CAPITULO

3

DESCRIPCIÓN DE LA SOLAR CONCENTRATION TOOLBOX

➢ *Resumen:*

Se describe detalladamente en este apartado la toolbox:

▪ En primer lugar, en el apartado 3.1, se resume de manera muy general el funcionamiento conjunto de la rutina y la estrategia de optimización seguida.

▪ En segundo lugar, apartado 3.2, se relacionan todas las subrutinas empleadas, ordenadas jerárquicamente en forma de esquema, y también se presenta una segunda relación con una breve descripción de la tarea que realiza cada una, con el fin de presentar una visión global y rápida del programa.

▪ Por último, en el apartado 3.3 se describen con detalle las funciones de manera individual siguiendo el siguiente esquema:
primero se describe la función principal de optimización; es la que engloba a todas las demás y las va llamando directa o indirectamente;
se seguirá con la función de mallado, las de pérdidas, y la de cálculos anuales, y por último, las de cálculos auxiliares y representación de los resultados.
En cada una de ellas se enumerarán: argumentos, cálculos que realizan con ellos, y parámetros que devuelven.

3.1) Software para el diseño y optimización de plantas en campos complejos

Se ha comentado en un apartado anterior que el software se ha desarrollado para el diseño y la optimización de plantas en *campos complejos.* Por campo complejo, entendemos, todo aquel campo que tenga una parcela de forma irregular que imponga unas restricciones de terreno. Estas restricciones de terreno pueden ser sólo perimetrales, el caso de una parcela plana cuya forma siga un polígono irregular determinado, cómo superficial, tenemos un polígono irregular en la ladera de una colina. Este último aspecto es especialmente importante, ya que es el factor diferencial principal respecto a la mayoría de los códigos existentes hasta ahora, incapaces de realizar cálculos de campos tridimensionales.

En el Capítulo 1 se ha reseñado que se ha puesto a punto la SCT, librería de funciones de Matlab, desarrollada por Marcelino Sánchez, en el contexto del proyecto SIREC. Esta puesta a punto ha requerido un gran trabajo, puesto que en la librería de partida ocurrían dos cosas; en primer lugar no estaba ensamblado el programa completo para parcelas no horizontales, y además la función existente para mallados no planos se ha desechado por no dar buenos resultados, implementándose una nueva técnica de mallado, basada en el artículo

de Siala y Elayeb [9]; en segundo lugar, ha habido que corregir varias funciones que no funcionaban correctamente con los ejemplos probados, en concreto, las de cálculos de sombras y bloqueos, además de los cambios necesarios para que el programa completo de la optimización se realize de manera adecuada a un cálculo no plano. En el apartado 3.2 se señalan con un asterisco las funciones que han sido modificadas y con dos las que se han creado y añadido a la librería.

Pero ¿Cómo funciona este software? ¿Cuál es su dinámica?

Se ha creado una función principal, **optimiza3D,** que es la que realiza el diseño, la simulación y optimización de la planta solar. Esta función realiza su tarea de la forma siguiente:

A partir de una altura máxima y mínima de torre y de unos ángulos máximo y mínimo de inclinación del receptor y con una potencia dada en el punto de diseño, el sistema crea una serie de mallados distintos; con estos mallados se calcula el comportamiento en el punto de diseño y se van escogiendo los helióstatos hasta que se alcanza la potencia requerida en dicho punto, así se tienen una serie de campos, pero ya restringidos por el propio terreno y por el punto de diseño; el siguiente paso es hacer los cálculos anuales, para ver cual de todas las opciones posibles es la mejor, esto se hace de manera recurrente entre los distintos campos posibles, las distintas alturas de torre y las distintas inclinaciones de receptor.

Optimiza3D necesita numerosas subrutinas para realizar el diseño de la planta: es fundamental la subrutina de mallado, **malla3D** que se ha programado en base a un método gráfico que dispone los helióstatos para que no haya bloqueos; adaptándolo y generalizando para geometrías complejas y terrenos inclinados; lógicamente también puede ser usado para un caso plano de forma muy sencilla. También son necesarias las funciones para cálculos de sombras, bloqueos, spillage y demás pérdidas, aunque hay que señalar que no se han tenido en cuenta algunas, como la absortividad y reflectividad de los helióstatos y del receptor, o las sombras provocadas por la torre. Una vez evaluadas las pérdidas en el punto de diseño hay que realizar los cálculos anuales; la función **performance3D** calcula el comportamiento anual de la planta solar determinado previamente en el punto de diseño por **optimiza3D** y los datos y parámetros geométricos introducidos como datos por el usuario.

¿Podría hacerse esta simulación de otra manera?

Se podrían definir otras estrategias de optimización; solo se debe definir bien la que se vaya a seguir y llamar a las diferentes subrutinas de forma adecuada, pero en principio no hay restricciones.

¿Qué ventajas tiene sobre los softwares existentes? Quizás las ventajas más importantes son la versatilidad, cualquier idea innovadora en el campo de la energía solar térmica puede ser incorporada, y evaluada en un tiempo razonable y la posibilidad de realizar cálculos tridimensionales y obtener resultados en terrenos inclinados, lo que no era posible hasta ahora.

3.2) Esquema y jerarquía del programa.

La función principal va utilizando las diferentes subrutinas que necesita para sus cálculos según, el esquema que se muestra a continuación:

- **Optimiza3D**
  - ★ **solpos**
  - ★ **cosdirinc***
  - ★ **malla3D** **
    - → ispar**
    - → rangoangular**
      - ? *cart2pol* [1]
      - ? intercirseg**
    - → ordena*
    - → *inpolygon*
  - ★ **vecinos3D***
  - ★ **atenuacion**
  - ★ **vértices***
    - → anguloreflex
    - → normalinreflex
    - → azimelev*
    - → giromatriz2*
    - → giromatriz3*
  - ★ **proyesfera**
  - ★ **sombrasplano***
    - → planocm
    - → xcorteycorte*
    - → solapevecinospm*
  - ★ **bloqueosesfera***
    - → proyesfera
    - → solapevecinoses*f
      - ? azimelev*
      - ? solapevecinospm*
  - ★ **spillage**

[1] Las funciones en cursiva pertenecen a la librería de funciones propia de Matlab, pero se reseñan aquí por su especial importancia. Se usan más funciones de Matab, de tipo matemático fundamentalmente pero solo se reseñarán las que tengan una importancia mayor para el funcionamiento del programa.

- ★ **sombraterreno****
- ★ **minpendiente****
- ★ **performance3D***
  - → vecinos3D*
  - → solpos
  - → cosdirinc*
  - → atenuación
  - → vértices*
    - ? anguloreflex
    - ? normalinreflex
    - ? azimelev*
    - ? giromatriz2*
    - ? giromatriz3*
  - → proyesfera
  - → sombrasplano*
    - ? planocm
    - ? xcorteycorte*
    - ? solapevecinospm*
  - → bloqueosesfera*
    - ? proyesfera
    - ? solapevecinosesf *
      - ▪ azimelev*
      - ▪ solapevecinospm*
  - → sombraterreno**
- ★ **costetorre**
- ★ **costereceptor**
- ★ **costeterreno****
- ★ **pintasolucion3D****

A continuación, una vez establecida la jerarquía del programa, se relaciona una pequeña explicación de cada una las funciones, *ordenadas alfabéticamente.*

FUNCION	TAREA
anguloreflex.m	Calcula los ángulos de reflexión de los rayos incidentes en los helióstatos
atenuación.m	Calcula la atenuación atmosférica debido a la distancia del helióstato al receptor
azimelev.m	Calcula los ángulos azimutal y de elevación a partir de los cosenos directores de los rayos incidentes
bloqueosesfera .m	Calcula los bloqueos usando proyecciones sobre una esfera alrededor del aiming point
bloqueosplano.m	Calcula los bloqueos usando proyecciones sobre un plano
cosdirinc.m	Calcula los cosenos directores de los rayos incidentes a partir de su azimut y su elevación
costereceptor	Calcula el coste del receptor en función de su área
costetorre.m	Calcula el coste de la torre en función de su altura
costeterreno	Calcula el coste del terreno en función de su área
giromatriz2.m	Proporciona una matriz que gira los helióstatos según dos ángulos
giromatriz3.m	Proporciona una matriz que gira los helióstatos según tres ángulos
intercirseg.m	Calcula la intersección de un círculo con un segmento, situados en cualquier posición
ispar.m	Calcula si un número entero dado es par o impar
malla3D.m	Crea mallados complejos y tridimensionales para las posiciones iniciales de los helióstatos siguiendo las ecuaciones adaptadas de delsol

FUNCION	TAREA
minpendiente.m	Calcula la pendiente límite para que no aparezcan sombras en el terreno
normalinreflex.m	Calcula la normal a una superficie sabiendo sus rayos incidente y reflejado
optimiza3D	Función principal para el cálculo, simulación y optimización de campos con geometría compleja
ordena.m	Ordena los helióstatos calculados según el método de no bloqueos
performance3D.m	Calcula el número total de kW que van a parar al receptor en los días y a las horas escogidos para representar el año
pintasolución3D.m	Función que pinta los resultados del campo óptimo a partir del struct array SOLUCION de optimiza3D.m
planocm.m	Calcula el punto de corte entre rectas y planos de forma matricial
proyesfera.m	Calcula las proyecciones de superficies planas sobre una esfera
rangoangular.m	Calcula el rango máximo de ángulos de mallado para cada radio
solapevecinosesf.m	Calcula el solapamiento de las proyecciones de polígonos sobre una esfera
solapevecinospm.m	Calcula solapamientos de polígonos regulares
sombrasplano.m	Función de cálculo de sombras mediante proyecciones en un plano
sombraterreno.m	Función que establece si el terreno está en sombra o no
solpos.m	Calcula la posición del sol

FUNCION	TAREA
spillage.m	Calcula las pérdidas debidas al “spillage” geométrico
vecinos3D.m	Encuentra los n vecinos (helióstatos) más próximos en el campo
vertices.m	Calcula las coordenadas de los vertices de los helióstatos para sus posiciones óptimas
xcorteycorte.m	Calcula los valores de las coordenadas x e y en el plano de corte necesario para sombras y bloqueos

Se introduce aquí una pequeña explicación del cometido de las dos funciones de Matlab reseñadas como más importantes.

FUNCION	TAREA
cart2pol	Función que transforma coordenadas cartesianas en coordenadas polares
inpolygon	Función que es capaz de evaluar si un punto dado por sus coordenadas *x*, *y* está dentro fuera o justo en el límite de un polígono

3.3) Descripción de las funciones

Función optimiza 3D

Función principal o programa que realiza el cálculo y la optimización de la planta solar

```
function[MAXPOW,MINCOST,THTOPTIM,TILTOPTIM,INDSOLUCION,SOLUCION]=
optimiza3D(THTMIN,THTMAX,TILTMIN,TILTMAX,DESIGNPOW,DESIGNDAY,
DESIGNHOUR,Ndays,Nhours,RADIACION,LATITUD,MEANFLUX,SIGMA,ALTURAH,
ANCHURAH,PEDESTH,RTOWER,XV,YV,CRITERIO)
```

argumentos :

THTMIN,THTMAX	*rango de alturas de torre para la optimización (metros)*
TILTMIN,TILTMAX	*rango de ángulos de inclinación del receptor respecto a la vertical (grados)*
DESIGNPOW	*potencia elegida para la planta en el punto de diseño (megawatios)*
DESIGNDAY	*día elegido para el punto de diseño*
DESIGNHOUR	*hora elegida por el punto de diseño*
Ndays	*días del año en los que se evalúan los cálculos anuales*
Nhours	*horas del día en las que se evalúan los cálculos anuales*
RADIACIÓN	*densidad de radiación solar en la zona (kW/m^2)*
LATITUD	*coordenada geográfica (grados)*
MEANFLUX	*flujo medio solar en el receptor (kW/m^2)*
SIGMA	*desviación estandar*
ALTURAH, ANCHURAH, PEDESTH	*dimensiones de los helióstatos y altura de su pedestal (metros)*
RTOWER	*radio de la torre (metros)*
XV, YV	*vértices de la parcela*

CRITERIO	*elección para la optimización: puede maximizar la potencia sin tener en cuenta los costes o puede minimizar la relación coste-potencia*

Los argumentos de esta función son los datos de entrada del programa. Si en un futuro se desarrollara una interface de usuario del mismo, estos serían los datos que habría que introducir para ejecutarlo.

Los argumentos de las demás funciones son resultados de cálculos del programa, por lo que no pueden ser fijados por el usuario.

Por otra parte, podrían señalarse como más relevantes y "propios" de **optimiza3D** los argumentos que hacen referencia al proceso de optimización: rango de alturas y ángulos, y criterio a elegir; los demás argumentos son usados por otras funciones que son llamadas por **optimiza3D**.

parámetros de salida:

MAXPOW	*máxima potencia media anual, obtenida por la planta con el campo óptimo (megawatios)*
MINCOST	*mínimo coste posible del sistema, para el campo óptimo (€/kilowatios)*
THTOPTIM	*altura óptima de la torre (metros)*
TILTOPTIM	*ángulo de inclinación óptimo para el receptor (grados)*
INDSOLUCION	*índice de la variable solución donde se encuentra el campo óptimo*
SOLUCION	*struct-array donde se almacenan todos los resultados medios anuales (pérdidas, nº de helióstatos, maxpotencia...) de todos los campos calculados*

Los parámetros de salida de esta función constituyen los resultados de salida del programa.

Se obtendrá un valor para MAXPOW cuando se haya elegido esta opción para la variable CRITERIO, mientras que MINCOST tendrá sentido cuando se tome la otra opción. En SOLUCIÓN se almacenarán los cálculos de todos los campos que sean capaces de proporcionar la potencia de diseño, no solo el óptimo.

descripción:

? El primer grupo de sentencias (ver código de función **optimiza3D** en *Anexo I*) es de inicialización de los argumentos de entrada, que como ya se ha comentado, son datos fijados para la planta por el diseñador, así como una serie de variables que actuarán como contadores en el proceso de comparación de la optimización.

? El segundo grupo es el de cálculos de la optimización.

En primer lugar, se toma un intervalo de alturas de torres dado por un valor máximo, uno mínimo y el intermedio entre los dos; de igual manera, se toman los valores para los ángulos de inclinación del receptor. Por tanto, en el primer intento hay tres alturas y tres ángulos, es decir, nueve posibilidades. De esta forma THT y TILT se irán variando para comprobar qué combinación de ellas da mejor resultado. El resto de los datos de entrada no son variables de optimización del problema y se mantendrán fijos para todos los casos.

Se llama a la función **solpos** que proporciona los ángulos de azimut y elevación de los rayos solares y a **cosdirinc** que transforma estos ángulos en cosenos directores, que son necesarios como argumentos de funciones que se utilizan más adelante.

A continuación, se realizarán los cálculos para el punto de diseño. Para ello hay que calcular la disposición del campo de helióstatos y las pérdidas de cada uno de ellos; ambos cálculos dependen de la altura de la torre, por lo que se tendrán que evaluar cada vez que cambiemos ésta. El campo lo proporciona la función **malla3D***.* Las pérdidas por atenuación, efecto coseno, sombras y bloqueos **son independientes del ángulo de inclinación del receptor**, no es necesario calcularlas cada vez que varíe el ángulo. Esto se consigue fijando la altura de torre y realizando el bucle de variación del ángulo posteriormente al cálculo de dichas pérdidas; para cada inclinación se calcularán las pérdidas por spillage, que **si dependen de este ángulo**. Cada una de las pérdidas se calculan con su correspondiente función**: atenuación, sombrasplano, bloqueosesfera y spillage***.*

Una vez evaluadas las pérdidas para cada helióstato, en tanto por uno, se obtiene la potencia total de la planta en el punto de diseño. Se comprueba si se consigue alcanzar la potencia de diseño:

-si se alcanza se continúa el proceso de cálculo

-si no se alcanza, se prueba con un nuevo ángulo y/o altura según corresponda.

Alcanzada la potencia requerida se “recortan” los helióstatos de peor rendimiento que sobren y se realizan los cálculos anuales mediante la función **performance 3D,** contabilizándose el intento; en la variable SOLUCION se guardan la potencia media anual obtenida y las pérdidas (ya en kW) medias anuales para cada campo; por otra parte en la variable SOLUCIONDESIGN se guardan la potencia y las pérdidas en el punto de diseño también para cada campo probado.

Ahora se evalúa el CRITERIO elegido y se bifurca el programa en dos ramas en paralelo, una para cada criterio:

→ Si se ha tomado la opción de **máxima potencia**, simplemente se evalúa si la potencia calculada para unos nuevos valores de THT y TILT es mayor que la anterior; si es mayor se guardan la altura y el ángulo correspondientes y si es menor se pasa directamente al siguiente grupo.

→ Si se ha tomado la opción del **mínimo coste**, primero hay que calcular el coste del caso actual; una vez obtenido se compara con el del anterior y si es menor el actual se almacenan su altura y su ángulo y si es mayor se pasa directamente al siguiente grupo, dejando los valores que hubiera.

? En el tercer grupo de órdenes, una vez que se evalúan las 9 primeras posibilidades se comprueba si merece la pena seguir optimizando, es decir, hacer otro grupo de 9.

Lo primero es ver si se ha elegido el caso de máxima potencia o de mínimo coste, ya que hay que comprobar si la máxima potencia obtenida del grupo de nueve actual es mayor que la del anterior e igualmente si el mínimo coste actual es menor que el del anterior.

Esta comprobación siempre se hará la primera vez, tras el primer grupo de 9, porque se ponen como límites, valores inalcanzables de potencia y de coste. Por tanto, como mínimo se evalúan 18 posibilidades.

En el caso de que haya que mejorar, se establecen las nuevas condiciones de contorno de altura e inclinación.

Las nuevas condiciones se eligen refinando el intervalo entorno al valor óptimo obtenido, por ejemplo, si el óptimo anterior fuera el extremo inferior del intervalo se modifica el valor superior del mismo, asignándosele el valor mínimo más un cuarto de la "longitud" inicial del intervalo; se tiene entonces un intervalo cuatro veces más pequeño que el inicial y además en torno al valor más óptimo. Lógicamente se contemplan los tres casos posibles: que el óptimo fuese el valor más pequeño, el más grande o el intermedio del intervalo inicial.

Con el nuevo intervalo se realizan otra vez todos los cálculos anteriores: un nuevo campo, unas nuevas pérdidas, etc, y todo ello para las 9 nuevas combinaciones.

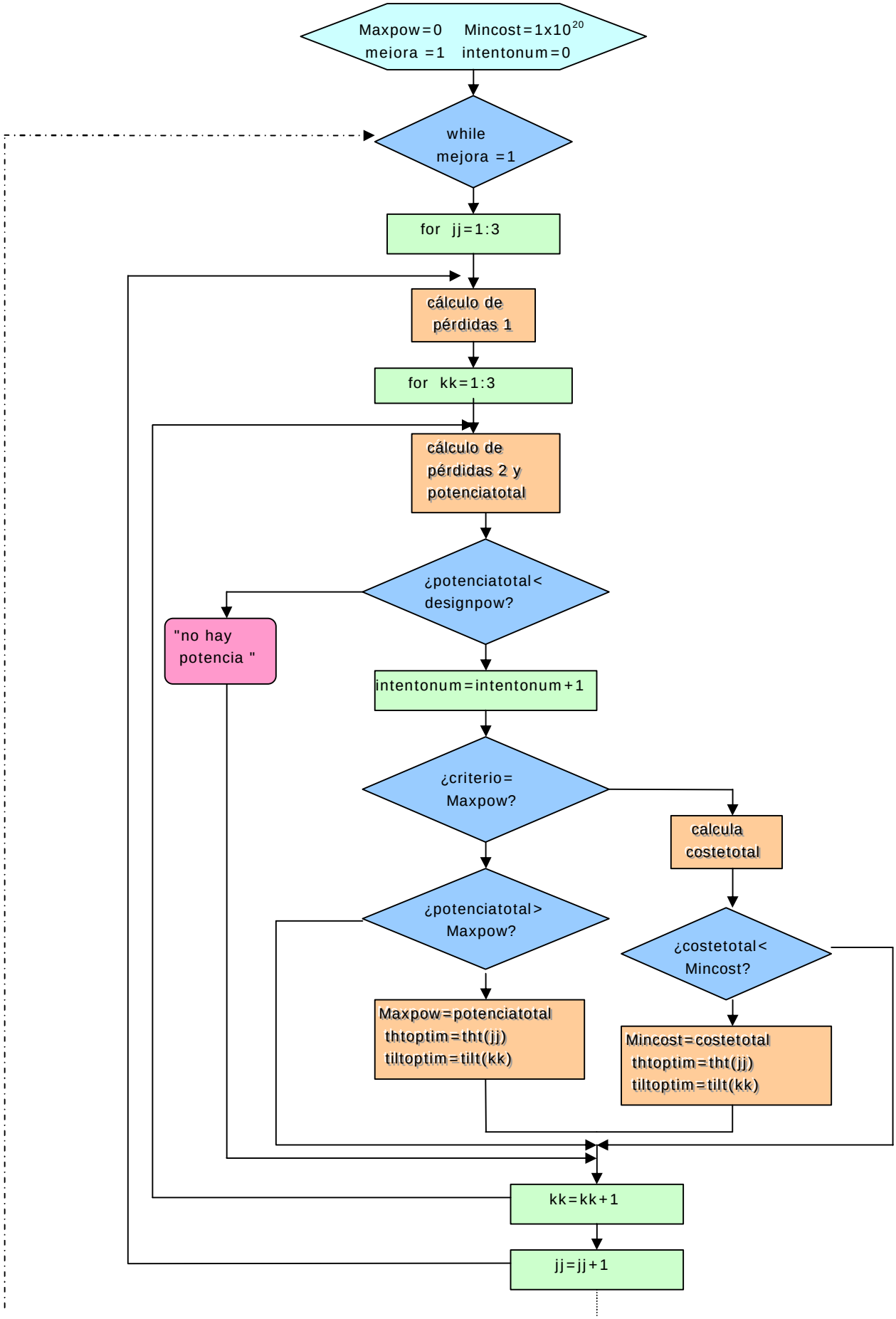
Si el valor máximo de potencia que se obtiene es mayor que el del grupo anterior (o en el caso de la opción coste, se consigue un coste menor) se sigue con el proceso de mejora y se evaluaría un nuevo intervalo.

En caso contrario, cuando ya no hay *mejora* se termina el proceso de optimización.

En la variable *SOLUCIÓN* queda reflejado el índice de cual de todos los campos probados es el óptimo, para así localizarlo fácilmente.

Por último, se llama a la función **pintasolución 3D** que representa gráficamente el campo óptimo final con sus pérdidas y proporciona también un fichero con los datos del mismo.

? Para una visión más clara del programa se adjunta a continuación un diagrama de flujo del mismo, que también se realizará en otras funciones, cuando sean de código complicado. También puede ayudar a la comprensión del funcionamiento del programa, el código del mismo que aparece en el *Anexo I* al final del trabajo.


Maxpow=0 Mincost=1x10^20
mejora =1 intentonum=0
while
mejora =1
for jj=1:3
cálculo de
pérdidas 1
for kk=1:3
cálculo de
pérdidas 2 y
potenciatotal
¿potenciatotal<
designpow?
"no hay
potencia "
intentonum=intentonum+1
¿criterio=
Maxpow?
calcula
costetotal
¿potenciatotal>
Maxpow?
¿costetotal<
Mincost?
Maxpow=potenciatotal
thtoptim=tht(jj)
tiltoptim=tilt(kk)
Mincost=costetotal
thtoptim=tht(jj)
tiltoptim=tilt(kk)
kk=kk+1
jj=jj+1

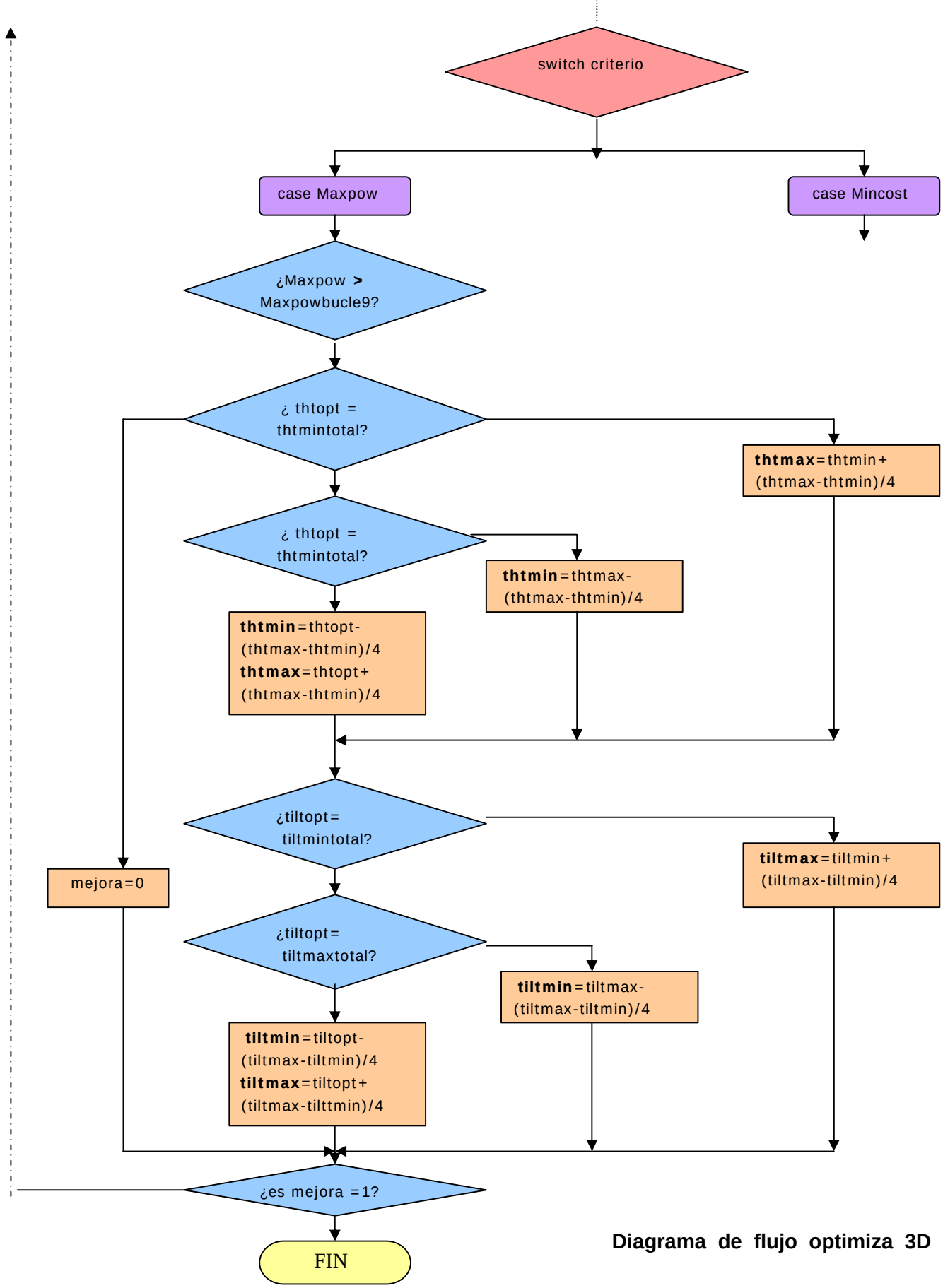


Diagrama de flujo optimiza 3D

Función malla 3D

Función llamada por **optimiza 3D**, que realiza el cálculo del mallado, es decir, determina las coordenadas *x, y, z* de los centros de los helióstatos.
Antes de nada decir que se han desarrollado dos tipos de funciones **malla 3D:**

1) Para el caso en que tenemos un terreno en forma de cono
2) Cuando el terreno es un plano inclinado

Expondremos la explicación genérica del mallado de helióstatos con la función del caso 1) y luego haremos las particularizaciones pertinentes para el caso 2).

1) Caso del cono

```
function[CAMPO3d,angulonuevo]=malla3D(HT,LM,WM,XV,YV,Z0,BL)
```

***argumentos*:**

HT	*altura de la torre (metros)*
LM, WM	*dimensiones de los helióstatos (metros)*
XV, YV	*vértices de la parcela (metros)*
Z0	*altura del pedestal de los espejos (metros)*
$ß_L$	*ángulo de inclinación de la parcela (grados)*

Todos los argumentos de esta función son parte de los datos fijados en **optimiza 3D** para el diseño de la planta.

***parámetros de salida*:**

CAMPO3d	*matriz de coordenadas x, y, z de las posiciones de los centros de los helióstatos*
angulonuevo	*ángulo de la parcela o terreno, medido respecto al plano horizontal*

descripción:

? La descripción de esta función, esencial para el funcionamiento del programa se estructura en dos bloques: primero se explican brevemente los pasos del método; segundo se formulan matemáticamente estos pasos.

? Para la programación de esta función se ha tomado como base un método gráfico para disponer los helióstatos. Dicho método elaborado por Pylkkanen [2] propone una disposición en radios alternados, calculándose los incrementos de radios y de ángulos para posicionarlos de manera que no existan pérdidas por bloqueos en la media anual. Aunque el método asegura que los helióstatos vecinos no se bloquean, no toma en consideración la posibilidad de que quede en sombra parte o incluso todo un helióstato. Esto se ha basado en la experiencia de que los bloqueos tienen un efecto más pronunciado que las sombras en la salida del campo. [7]

Es un método sencillo comparado con los procedimientos de celdas utilizados por DELSOL. Se representa por un conjunto de ecuaciones matemáticas, lo que facilita su implementación en el lenguaje de programación correspondiente.

Para llevar a cabo el mallado se descomponen los posibles bloqueos entre helióstatos en un bloqueo en la dirección azimutal y un bloqueo en dirección vertical de tal forma que si uno de ellos es nulo, se asegura la inexistencia del bloqueo total.

El procedimiento de mallado puede resumirse en los siguientes pasos, y se ilustra con la ayuda de la **Figura 3.1** en la siguiente página:

1) Situar las dimensiones del campo en el papel, con el *radio mínimo* del campo igual a 0,75 veces la altura de la torre.

2) Cada helióstato es representado por un círculo cuyo diámetro en planta sea igual a la diagonal del helióstato más una cierta distancia, lo que permitirá la colocación del siguiente anillo *"al tresbolillo"* como se explica en el siguiente apartado. En alzado el diámetro será igual a la altura del helióstato (para evitar bloqueos verticales).

3) El espaciado azimutal entre helióstatos, en el primer anillo de cada grupo se toma igual a dos veces el ancho del helióstato como mínimo. De esta forma se pueden colocar los siguientes anillos del grupo en posición *"al tresbolillo"* con la misma separación azimutal y evitando así los bloqueos totales entre anillos consecutivos al hacer nulos los bloqueos de la dirección azimutal. El radio del segundo anillo (alternado) en cada grupo se determina como la distancia más cercana al primer anillo que evite las interferencias geométricas(es decir, el choque) entre el primer y el segundo anillo.

4) El radio del tercer anillo del mismo grupo se determinará para que los bloqueos verticales con respecto al primer anillo del grupo sean nulos. No es necesario tener en cuenta para nada el segundo anillo ya que la posición *"al tresbolillo"* nos asegura bloqueo azimutal nulo. Se seguirán los siguientes pasos para determinar el tercer radio:

a) Se representa el helióstato ya situado, del anillo anterior por el círculo C_1 como se explicó en 2).

b) Desde el punto *a* se traza una línea recta que sea tangente al círculo C_1 en el punto *d*. La línea *ab* representa el límite inferior de los rayos reflejados detrás del helióstato C_1 y que llegan al receptor sin ser bloqueados por C_1

c) Se dibuja un círculo C_2, que representa al helióstato en el anillo en que se vaya a situar, de manera que sea tangente a *ab* en el punto *e*.

d) Se localiza el centro de C_2, que fijará el radio del anillo que se quería determinar.

5) Una vez hallado el tercer radio es interesante decidir si empezando un nuevo grupo aumentaría la densidad de helióstatos en el terreno. Se puede explicar este fenómeno de la siguiente forma:

a) Si se mantiene un mismo grupo, la posición *"al tresbolillo"* hace que al ir añadiendo radios a ese mismo grupo los heliostastos se separen excesivamente en la dirección azimutal.

b) Si se cambia de grupo, empieza un nuevo *"tresbolillado"* de máxima compactación azimutal en el primer anillo, pero a cambio tendríamos que separar, en dirección radial, el primer anillo de este segundo grupo del último anillo del anterior grupo, de forma que no hubiera bloqueo vertical entre ellos, ya que no se asegura el bloqueo azimutal. En el caso a) sin embargo, esta separación radial era respecto al penúltimo anillo del grupo y por lo tanto menor, debido a que se tenía asegurado el bloqueo azimutal respecto al último anillo (por la posición *"al tresbolillo"*).

6) Para el resto de anillos del grupo se seguiría el mismo procedimiento explicado en los pasos 4) y 5).

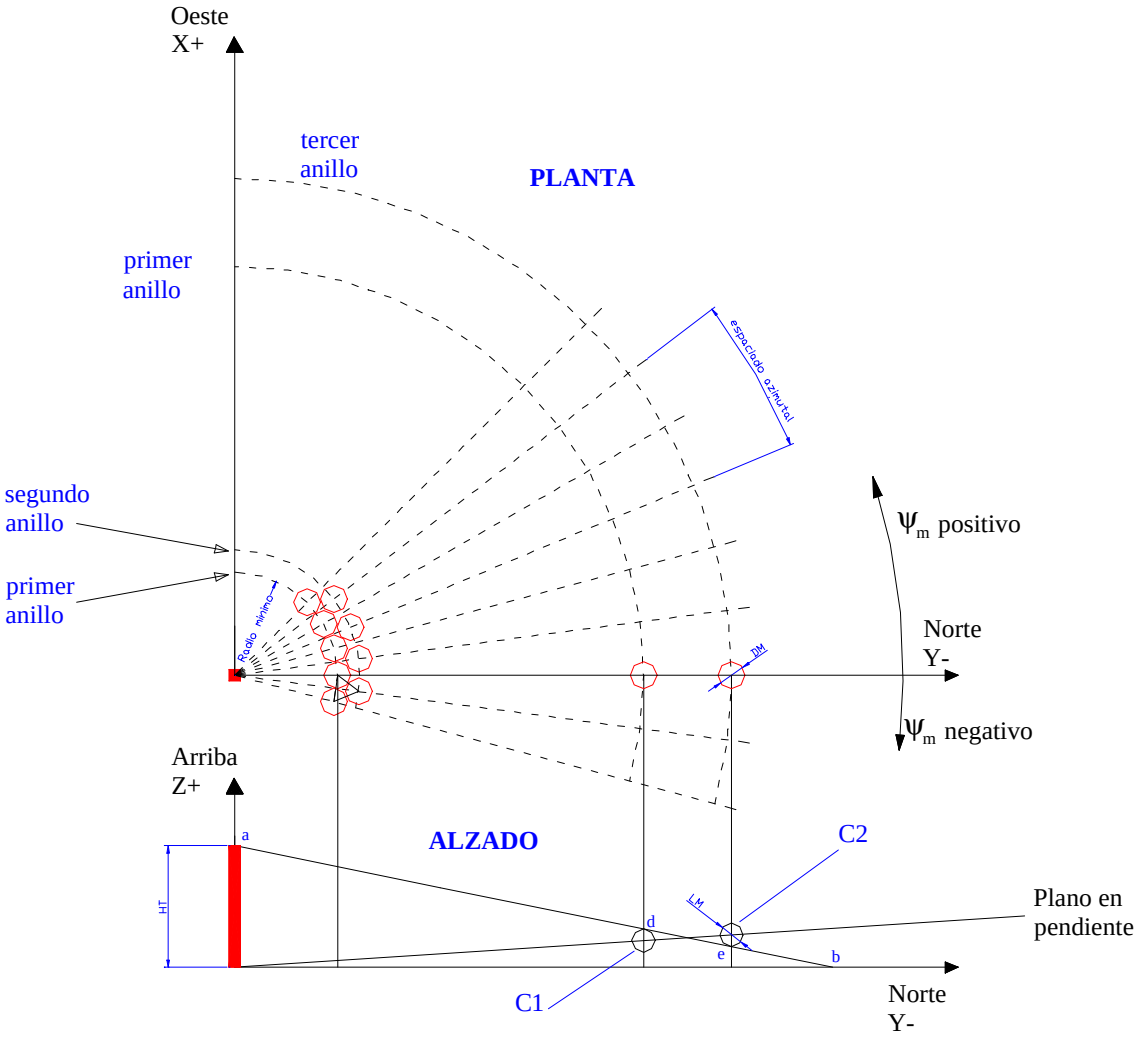


Figura 3.1: Disposición de los helióstatos, representados por círculos, en planta y alzado de manera que no existan bloqueos entre ellos. Se definen también el radio mínimo para el primer anillo y el espaciado azimutal entre helióstatos.

? A continuación, se expone la formulación matemática de los pasos explicados previamente. Para ello, se introducen las definiciones de algunos conceptos y para clarificar estas últimas se introducen algunas figuras.

Anillo centrado : Son aquellos que tienen un helióstato en el eje Y- del campo, es decir, en la dirección Norte.

Anillo alternado: Aquellos que no tienen helióstatos en la dirección Norte.

Diámetro característico (DM): Es igual a la diagonal del helióstato más una cierta distancia de separación:

$$DM = l_m \cdot \left(\sqrt{1+ f^2} + dS \right)$$

donde:

l_m, es el alto del helióstato

f, es el ancho partido el alto del helióstato

dS, es el cociente entre la distancia de separación entre helióstatos y el alto del mismo.

Para que no haya bloqueos, el valor mínimo de dS ha sido dado por Collado y Turegano [8] como:

$$dS_{min} = 2f - \sqrt{1+ f^2}$$

Por lo que sustituyendo en la definición de *DM*, obtenemos el valor de DM_{min}

$$DM_{min} = 2w_m$$

w_m es el ancho del helióstato

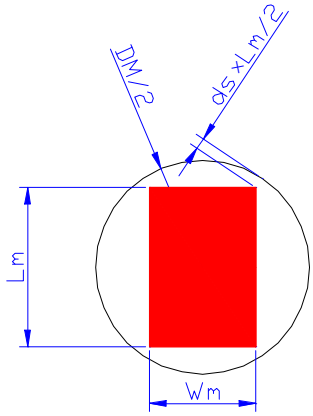


Figura 3.2: Dimensiones del helióstato y diámetro característico (DM).

Dirección angular unitaria: Ángulo entre los ejes de distribución de los helióstatos de cada grupo en radianes.

$$g_j = \frac{DM/2}{R_{0,j}}$$

Es una aproximación, ya que para obtener un ángulo en radianes hay que dividir entre el arco no por la cuerda del ángulo.

Grupo: Son los helióstatos que presentan una misma dirección angular unitaria, es decir una misma g_j . Un grupo siempre empieza con un anillo centrado

Dirección angular: Ángulo formado por el eje Y- del campo (dirección Norte) y los radios donde se sitúan los helióstatos dado por:

$$y_m = \pm n g_j$$

donde

n = 0, 2, 4...para anillos centrados
n = 1, 3, 5...para anillos alternados

➢ *Cálculo del espaciado azimutal*

El espaciado azimutal del primer anillo de cada grupo es el mínimo posible, que es igual a DM_{min}. El espaciado azimutal de los demás anillos del grupo depende de la divergencia de los ejes de distribución que está determinada por el espaciado azimutal del primer anillo.

➢*Cálculo del espacio entre radios*

El valor del radio de cada anillo viene determinado por el tipo de anillo.

El *primer anillo del campo*, que es por definición un anillo centrado, viene dado por distintas correlaciones en función de la altura de la torre; aquí se ha elegido la recomendada por Falcone en la que se toma:

$$R_{o,o} = R_{min} \cong 0.75 H_t$$

Para el *segundo anillo de cada grupo*, que por definición es alternado, el *radio medido en un terreno inclinado un ángulo ßL* viene dado por:

$$R'_{1,j} = R^i_{o,j} \cos g_j + \sqrt{(DM)^2 - (R_{0,j} sing_j)^2}$$

mientras que en el plano horizontal $R_{1,j}$ viene dado por:

$$R_{1,j} = R'_{1,j} \cos b_L = R_{o,j} \cos g_j + \sqrt{(DM \cos b_L)^2 - (R_{0,j} sing_j)^2}$$

En el campo CESA-1 el mínimo incremento de radio para evitar las interferencias geométricas(es decir, el choque) entre anillos, es igual a la altura de un triángulo equilátero de lado DM, por tanto

$$R_{1,j} \approx R_{o,j} + DM \cos 30^o \cos b_L$$

En ambas ecuaciones *DM* está medido en la superficie inclinada.

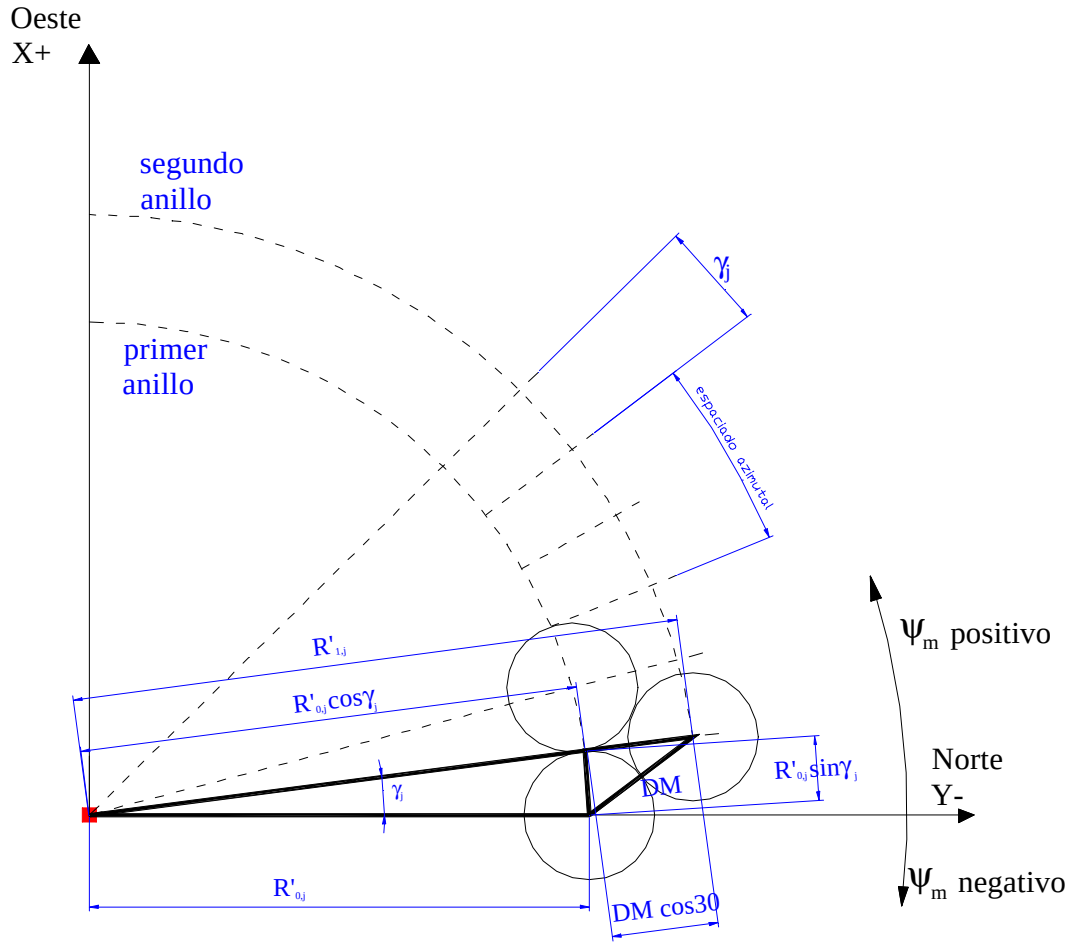


Figura 3.3: Cálculo del espaciado azimutal y del espaciado entre radios. El primero se determina con el cociente entre el ancho del helióstato y radio del primer anillo del grupo. El segundo mediante el cálculo de la altura de un triángulo equilátero de lado DM.

El radio de *cualquier otro anillo* se determina a continuación:

En la **Figura 3.4** se representa en alzado un helióstato conocido del anillo previo al radio que se está calculando. *ab* representa el límite más bajo para que los rayos reflejados originados por detrás de C_1 que alcanzan el receptor sin bloquearse con C_1 . El radio del nuevo anillo, $y_{m2,}$ se obtiene resolviendo las ecuaciones de la línea *ab* y los círculos C_1 y C_2

$$R_{i,j} = y_{m2} = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

$$A = -[(2H_T y_r + \tan b_L)\tan b_L + H_T^2 y_r^2$$

$$B = 2(H_T - z_0)(H_T y_r + \tan b_L)$$

$$C = (r_{m2})^2(1 + H_T^2 y_r^2) - (H_T - z_0)^2$$

$$y_r = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$a = H_T^2 (r_{m1}^2 - y_{m1}^2)$$
$$b = 2 y_{m1} H_T (H_T - z_{m1})$$
$$c = r_{m1}^2 - (z_{m1} - H_T)^2$$

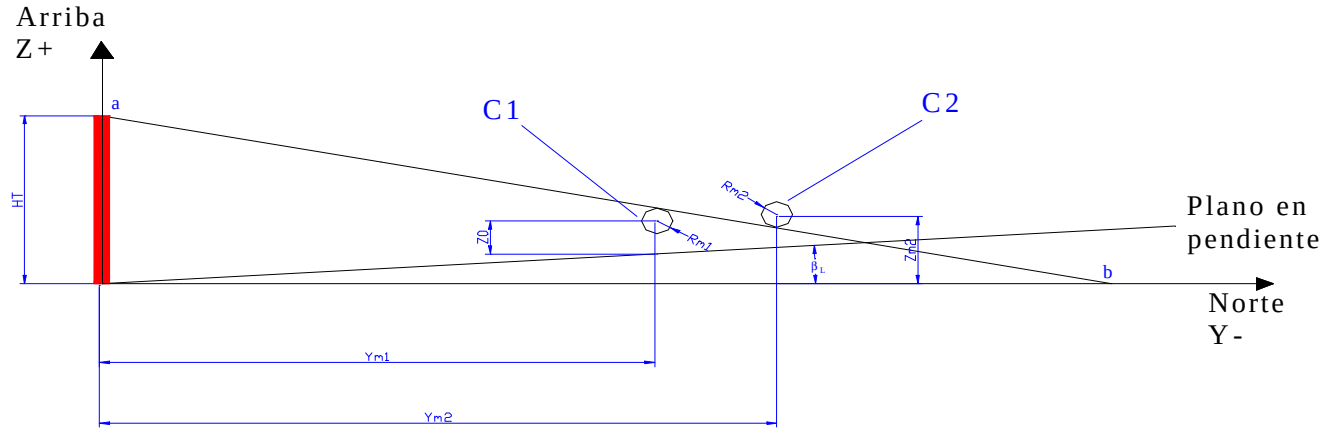


Figura 3.4: Cálculo del radio del tercer anillo en adelante, para cada grupo. Se determina resolviendo el sistema de ecuaciones formado por las ecuaciones de las circunferencias que representan los helióstatos de los radios contiguos y la de la recta ab

El radio de cualquier anillo en la superficie inclinada viene dado por: $R'_{i,j} = \frac{R_{i,j}}{\cos b_L}$

➢ *Cálculo de la posición del helióstato*

Un helióstato se sitúa en el campo definiendo las coordenadas de su centro, que coinciden en la vertical con las del pedestal, puesto que es el punto fijo de éste. Estas coordenadas son conocidas una vez que se ha determinado la dirección angular del helióstato y el radio del anillo al cual pertenece:

Estas fórmulas son válidas en el caso de un terreno en forma de cono.

$$x_m = R_{i,j} \sin \Psi_m$$
$$y_m = R_{i,j} \cos \Psi_m$$
$$z_m = z_o + R_{i,j} \tan b_L$$

➢ *El criterio de la densidad de espejo*

Se ha definido previamente el concepto de grupo, como el conjunto de anillos que presentan una misma dirección angular en posición *"al tresbolillo"*. Ya se ha comentado la necesidad de decidir el cambio ó no a un nuevo grupo, ya que el añadir un nuevo anillo a un grupo conlleva el aumento del espaciado azimutal, debido a la divergencia, que lógicamente aparece en la dirección angular. Por ello, cada vez que se añade un nuevo anillo al campo de helióstatos hay que tomar una decisión: añadir el anillo en el mismo grupo, o comenzar un nuevo grupo con una nueva dirección angular.

El criterio elegido para tomar la decisión anterior se fundamenta en el hecho de que hay que maximizar el área cubierta por superficie reflectante. Esta medida de aprovechamiento del terreno es la densidad de espejo, *d* definida como:

$$d = \frac{areanetadereflexión}{areadelterreno} = \frac{N_m A_m}{A_f}$$

Así la opción que proporcione un *d* mayor será la que se elija.

Señalar que el número de helióstatos se calcula de manera diferente para los anillos centrados y los alternados:

$$N_m = 2xentero\left(\frac{\Psi_{\max}}{2g} \right) + 1$$

$$N_m = 2xentero\left(\frac{(\Psi_{\max} - g)}{2g} \right) + 2$$

donde $g = g_j$ si se continúa con el mismo grupo y $g = g_{j+1}$ si se comienza con uno nuevo.

El área neta reflectante para cada helióstato viene dada por $A_m = l_m w_m f_a$ donde f_a es el cociente entre la superficie reflectante del helióstato y la superficie total del mismo.

El área de terreno que se evalúa para tomar la decisión viene dada por:

$$A_f = \Psi_{\max}\left[(R_{i+1,j} + 0.5DM)^2 - (R_{i,j} + 0.5DM)^2 \right]$$

2) Caso del plano

A continuación se desarrolla la explicación de la función **malla 3D** para el caso de un terreno en forma de pendiente plana.

Toda la metodología seguida en el caso anterior consistente en anillos de helióstatos centrados alrededor de la torre ha sido posible realizarla debido al hecho de que el terreno en forma de cono posee simetría de revolución respecto a dicha torre.

En el caso de terreno en forma de pendiente plana, dicha simetría no existe, por tanto una posible alternativa es realizar un cambio de coordenadas de forma tal que se haga el mallado de helióstatos centrando los anillos alrededor de una torre virtual. En estos nuevos ejes de coordenadas **(X', Y', Z')**, sí existiría ahora simetría de revolución del terreno (que ahora sería simplemente un plano horizontal) respecto a la torre virtual. Una vez acabado el mallado de helióstatos en los ejes relativos, sería necesario deshacer el cambio de coordenadas para volver al sistema de referencia global **(X, Y, Z)**.

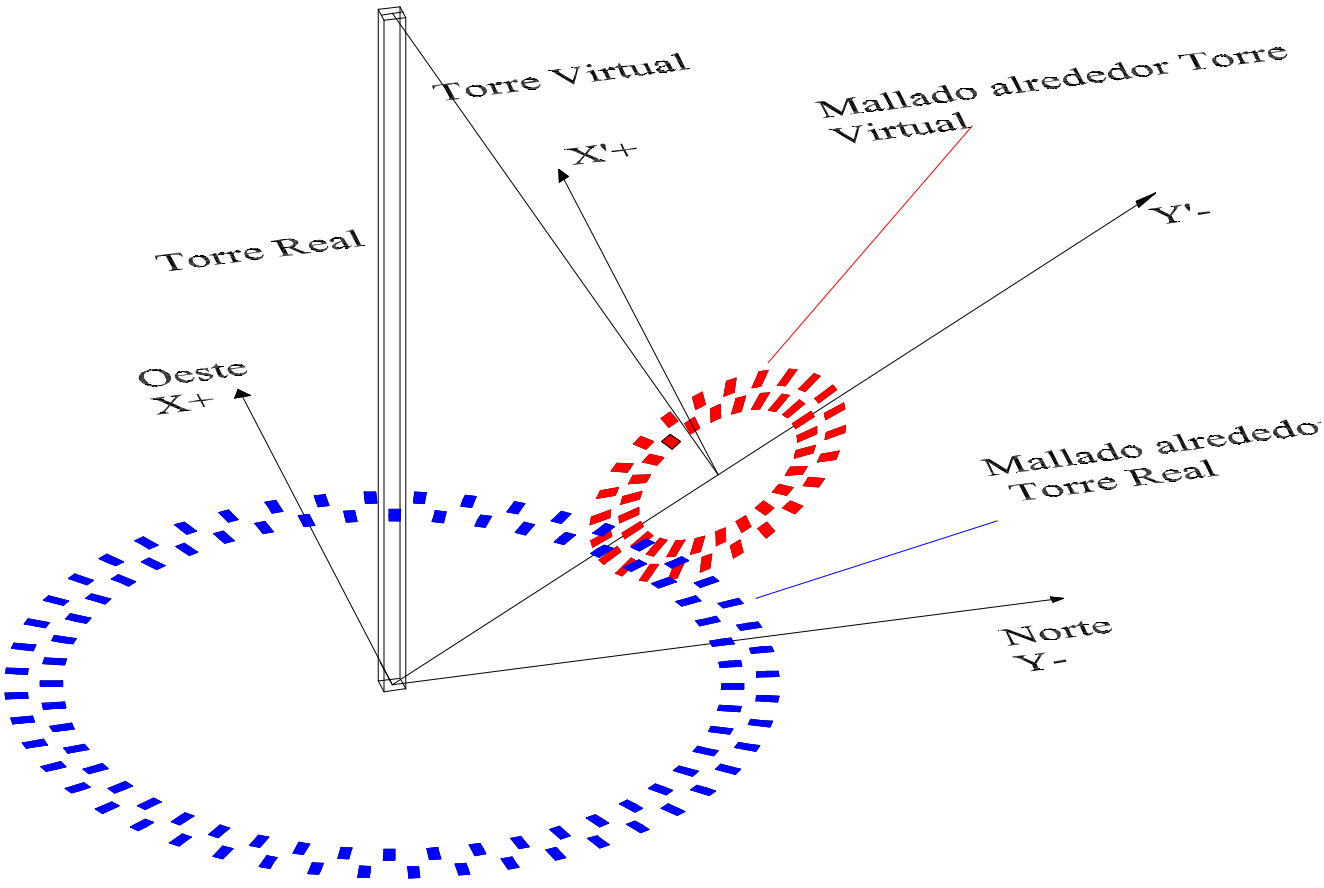


Figura 3.5: Cambio de coordenadas del sistema X, Y, Z asociado a una superficie plana, al sistema X',Y',Z' asociado a una superficie plana que forme un ángulo $ß_L$ con la horizontal.

```
function[CAMP,XCENTRO,YCENTRO,ZCENTRO,HT,Area]=malla3D(AlturaTorreReal,
LM,WM,XV,YV,ALTPEDES,Pendiente)
```

***argumentos*:**

AlturaTorreReal	*altura de la torre real (metros)*
LM, WM	*dimensiones de los helióstatos (metros)*
XV, YV	*vértices de la parcela (metros)*
ALTPEDES	*altura del pedestal de los espejos (metros)*
Pendiente	*ángulo de inclinación de la parcela (grados)*

Todos los argumentos de esta función son parte de los datos fijados en **optimiza 3D** para el diseño de la planta.

***parámetros de salida*:**

CAMP	*matriz de coordenadas x, y, z de las posiciones de los pedestales de los helióstatos*
XCENTRO, YCENTRO, ZCENTRO	*coordenadas de la torre tras el cambio de variables*
HT	*altura de la torre virtual (metros)*
Area	*área de la parcela o terreno dado por XV e YV medida en horizontal (metros cuadrados)*

descripción:

? La descripción de esta función sería análoga a la realizada para el terreno en forma de cono. Una diferencia significativa que se podría comentar sería el cálculo de la posición de los helióstatos que en este caso se haría:

$$x_m = R_{i,j} \sin \Psi_m$$
$$y_m = R_{i,j} \cos \Psi_m$$
$$z_m = z_o + R_{i,j} \tan b_L = z_o$$

donde $b_L = 0$ en este caso, ya que al trabajar en unos ejes de referencia relativos al terreno, éste es un plano horizontal. Posteriormente habría que deshacer el cambio de variables para volver al sistema de coordenadas global:

$$CAMP(:,1) = CAMPO3d(:,1) + XCENTRO$$
$$CAMP(:,2) = \cos(Pendiente).\times CAMPO3d(:,2) + \sin(Pendiente).\times CAMPO3d(:,3) + YCENTRO$$
$$CAMP(:,3) = -\sin(Pendiente).\times CAMPO3d(:,2) + \cos(Pendiente).\times CAMPO3d(:,3) + ZCENTRO$$

Este es el cambio de variables con el que se pasa de unas coordenadas a otras, como. se puede ver en el código de la función, en el *Anexo I* del proyecto.

Función atenuacion

Función llamada por **optimiza3D** y posteriormente por **performance3D** para calcular la atenuación atmosférica en el punto de diseño y la atenuación media anual respectivamente.

```
function [ATENUACION]=atenuacion(distanaim)
```

argumentos:

distanaim	*distancia del centro de cada helióstato del campo al aim point (metros).*

Este argumento es un vector de tamaño *nº de helióstatos x 1* ya que es el módulo de cada uno de los vectores que unen los centros de los helióstatos con el aim point *(vectoraim).* Es calculado en **optimiza3D.**

parámetros de salida:

ATENUACIÓN	*pérdidas por atenuación atmosférica de cada helióstato (tanto por 1)*

ATENUACIÓN es un vector, también de dimensiones *nº de helióstatos x 1*, porque obtenemos un valor de esta pérdida para cada helióstato.

descripción:

? En esta función se han modelado de manera “aparentemente sencilla” las pérdidas por atenuación. Físicamente la atenuación representa la pérdida de la radiación de los rayos solares por factores atmosféricos, debido a la distancia entre helióstato y receptor.

? Fundamentalmente, solo depende de esta distancia, que está modelada por *distanaim* (único argumento); pero también dependerá de las condiciones atmosféricas y visibilidad ya que lógicamente no habrá la misma atenuación en un día claro y en un ambiente limpio, que un día nublado o con polvo; también, aunque en menor medida, la estación del año o la altitud local afectan. [7]

La atenuación se modela como una función cúbica de *distanaim*, siendo los coeficientes de este polinomio cúbico unos valores tabulados que modelan las condiciones atmosféricas. [8]

El informe de Pitman y Vant-Hull presenta esta fórmula para estimar las pérdidas por atenuación entre helióstato y receptor. Es independiente de la longitud de onda y es un ajuste funcional a los resultados tabulados de Vittitoe y Biggs los cuales son el resultado de la integración numérica de datos de transmitancia espectral realizada con el código LOWTRAN 3. La formula permite la interpolación y la extrapolación y tiene el formato característico de un modelo de transmitancia atmosférica.[9]

? La función simplemente evalúa este polinomio con el valor de *distanaim* calculado y los coeficientes que se elijan en función de donde esté situada la planta.

$ATENUACION = \left(ATM1 + ATM2 \times R + ATM3 \times R^2 + ATM4 \times R^3\right)\frac{1}{100}$, donde:

ATM1, ATM2, ATM3, ATM4 coeficientes tabulados

R *distanaim/1000*

? La inclinación del terreno afectará a la atenuación a través de *distanaim*; parece evidente que si el terreno es inclinado tomará valores más pequeños que si es plano, al acercar los espejos al receptor, por lo que a igualdad de factores atmosféricos, las pérdidas por atenuación en un terreno inclinado deben ser menores que en un terreno plano.

Función sombrasplano

Función llamada por **optimiza3D** y por **performance3D** para calcular las pérdidas por sombras, en el punto de diseño y las medias anuales, respectivamente.

```
function[SOMBRASTOTAL,Areautil]= sombrasplano(cosinc,XN,YN,ZN,parejasvecinos)
```

argumentos:

cosinc	*vector cuyas componentes son los cosenos directores de los rayos solares incidentes (1x3)*
XN, YN, ZN	*matrices con las coordenadas x,y,z respectivamente, de los vértices de los helióstatos en su posición óptima (nº de helióstatos x 4)*
parejasvecinos	*matriz que indica los 'n' helióstatos más próximos a uno dado [(m x n) x 2] , donde 'm' es el número total de helióstatos.*

cosinc ha sido calculado previamente por **cosdirinc**, XN, YN, ZN por **vertices** y *parejasvecinos* por **vecinos3D**, siendo por tanto resultados de cálculos del programa que no pueden ser fijados por el usuario.

parámetros de salida:

SOMBRASTOTAL	*vector que indica el tanto por uno de sombra de cada helióstato (nº de helióstatos x 1)*
Areautil	*vector que indica el área del espejo "tras" el efecto coseno pero sin sombras (nº de helióstatos x 1)*

Con ellos se conocen pues, las pérdidas por sombra y por efecto coseno; para obtenerlas en kW basta con multiplicar por la radiación solar incidente (kW/m^2) y por el área total de helióstato.

descripción:

? Las pérdidas por sombras aparecen cuando el rayo de sol no llega al área de captación del helióstato al ser interceptado por un cuerpo opaco; normalmente se deben a los helióstatos vecinos y si la disposición del campo es correcta, solo deben aparecer cuando el sol esté bajo, es decir, al amanecer o al anochecer.

? El procedimiento para calcular las sombras comienza tomando un plano de referencia suficientemente alejado del sistema de coordenadas de trabajo sobre el que se proyectarán los helióstatos. Lo primero es posicionar este plano de referencia: se le asigna la distancia a la que se va a situar (por ejemplo 50 Km) y mediante los cosenos directores de los rayos solares se sitúa este plano perpendicular a dichos rayos en cada instante.

Ahora se proyectan en este plano los 4 vértices de todos los helióstatos del campo (dados por XN, YN, ZN). Para realizar esta proyección se llama a la funciones **planocm,** y **xcorteycorte**; de forma resumida, lo que hacen es calcular la intersección entre rectas que pasan por los vértices de los helióstatos (paralelas a los rayos del sol) y el plano imaginario de proyección, y como resultado proporcionan los vectores *xcorte* e *ycorte* que contienen las coordenadas de los cuatro vértices de todos los espejos proyectados en el plano de referencia.

Con *xcorte* e *ycorte,* y *parejavecinos* la función **solapevecinospm** calculará las superposiciones entre las proyecciones más próximas o vecinas, y como resultado se obtendrán las parejas que se hacen sombra, las áreas de los solapes que se producen, que lógicamente representan las pérdidas por sombra que se ocurren en el helióstato, y también las áreas útiles de cada uno de éstos.

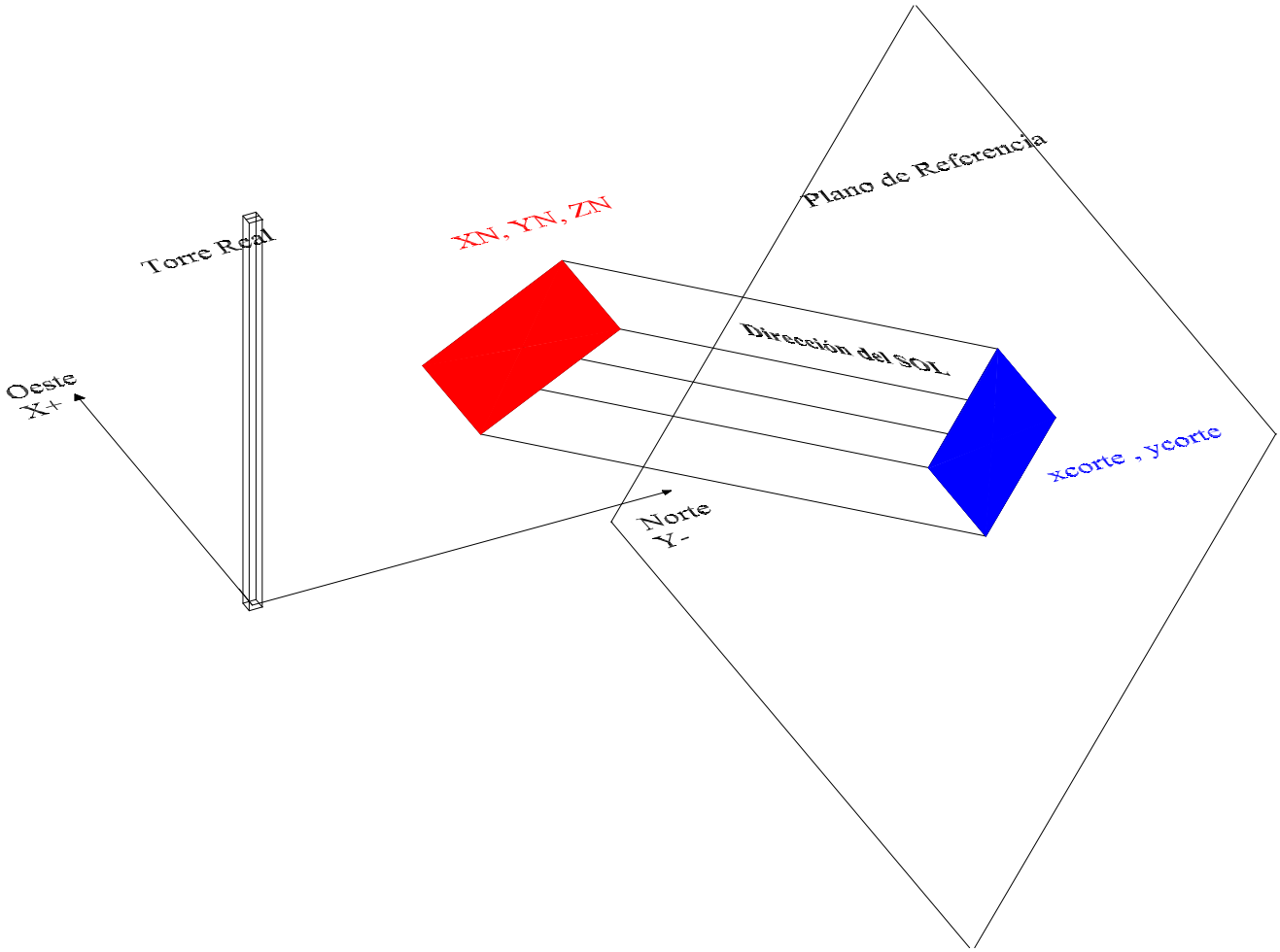


Figura 3.6: Procedimiento para calcular las sombras. Posición del plano de referencia en el que se proyectan las sombras de los helióstatos y se calculan sus solapamientos.

? Una vez conocidos los solapamientos en sombra de todas las parejas, se determina cual de los helióstatos de cada pareja hace sombra al otro; esto se hará viendo cuales son las distancias medias de cada espejo al plano de referencia anterior.

? Por último, calculamos las pérdidas porcentuales en los helióstatos, dividiendo los solapamientos por el área útil de cada uno de ellos, dando como resultado la variable *SOMBRASTOTAL .*

Función bloqueosplano

Función llamada por **optimiza3D** y por **performance3D** para calcular las pérdidas por bloqueos en el punto de diseño y las pérdidas medias anuales respectivamente.

```
function[BLOQUEOSTOTAL]=bloqueosplano(parejasvecinos,distanaim,vectoraim,
XN,YN,ZN,CAMPO)
```

argumentos:

parejasvecinos	*matriz que indica los 'n' helióstatos más próximos a uno dado [(m x n) x 2], donde 'm' es el número total de helióstatos.*
distanaim	*distancia del centro de cada helióstato del campo al aim point (nº de helióstatos x 1, metros) .*
vectoraim	*matriz de los cosenos directores de los rayos que unen los helióstatos con el aim point (nº de helióstatos x 3).*
XN, YN, ZN	*matrices con las coordenadas x, y, z respectivamente, de los vértices de los helióstatos en su posición óptima (nº de helióstatos x 4).*
CAMPO	*matriz con las coordenadas x, y, z del campo colector (nº de helióstatos x3).*

CAMPO es la matriz calculada por **malla3D.** XN, YN, ZN y *parejasvecinos* tienen iidéntico significado que para **sombrasplano**. *distaim* y *vectoraim* son utilizados de la misma manera que en **atenuación.**

parámetros de salida:

BLOQUEOSTOTAL	*vector que indica el tanto por uno de bloqueo de cada helióstato (nº de helióstatos x 1)*

Se determinan con este vector las pérdidas por bloqueos en cada espejo.

descripción:

? Las pérdidas por bloqueos ocurren cuando un rayo de sol reflejado por el área de un espejo es interceptado en su camino al receptor y no llega a éste, por causa de interposición de alguno de sus vecinos.

? El procedimiento para calcular los bloqueos es similar al de las sombras, pero repitiéndolo para cada grupo de helióstatos del vector *parejasvecinos.* Esto es necesario hacerlo porque ahora el plano de proyección sobre el que evaluaremos los solapamientos

cambiará para cada uno de estos grupos, ya que también lo hace la dirección de proyección (paralela al rayo reflejado desde cada helióstato al receptor).

? Para el primer grupo de vecinos, comenzaremos posicionando el plano de referencia en el centro del helióstato del grupo menos alejado del *aim point*, más 10 m en la dirección *vectoraim* del helióstato de referencia, para evitar problemas al proyectar. El plano será perpendicular a la dirección *vectoraim*.

Ahora se proyectan en este plano los 4 vértices de todos los helióstatos del grupo (dados por XNI, YNI, ZNI). Para realizar esta proyección se llama a las funciones **planocm,** y **xcorteycorte**; de forma resumida lo que hacen es calcular la intersección entre rectas que pasan por los vértices de los helióstatos (paralelas a *vectoraim* del helióstato de referencia) y el plano imaginario de proyección, y como resultado proporcionan los vectores *xcorteI* e *ycorteI* que contienen las coordenadas de los cuatro vértices de todos los espejos del grupo proyectados en el plano de referencia.

Con *xcorteI* e *ycorteI,* y *parejavecinos* la función **solapevecinospm** calculará las superposiciones entre las proyecciones del grupo de vecinos, y como resultado se obtendrán las parejas que se bloquean, las áreas de los solapes que se producen, que lógicamente representan las pérdidas por bloqueos que se producirán en el helióstato, y también las áreas de proyección de cada helióstato.Ver la **Figura 3.7** en página siguiente

? Una vez conocidos los solapamientos en bloqueos del grupo, se determina cual de los helióstatos de cada pareja bloquea al otro; esto se hará viendo cuales son las distancias medias de cada espejo al *aim point*.

? Por último, se calculan las pérdidas porcentuales en los helióstatos, dividiendo los solapamientos por el área de proyección de cada uno de ellos, dando como resultado la variable *BLOQUEOSTOTAL .*

? Este proceso se repetiría para cada grupo de vecinos del vector *parejasvecinos.*

? Es interesante comentar que este proceso es mucho más lento que el cálculo de las sombras, debido precisamente a que hay que ir cambiando el plano de proyección para cada grupo de vecinos. Por ello, se introduce a continuación una forma aproximada de computación de los bloqueos mucho más eficiente en este sentido.

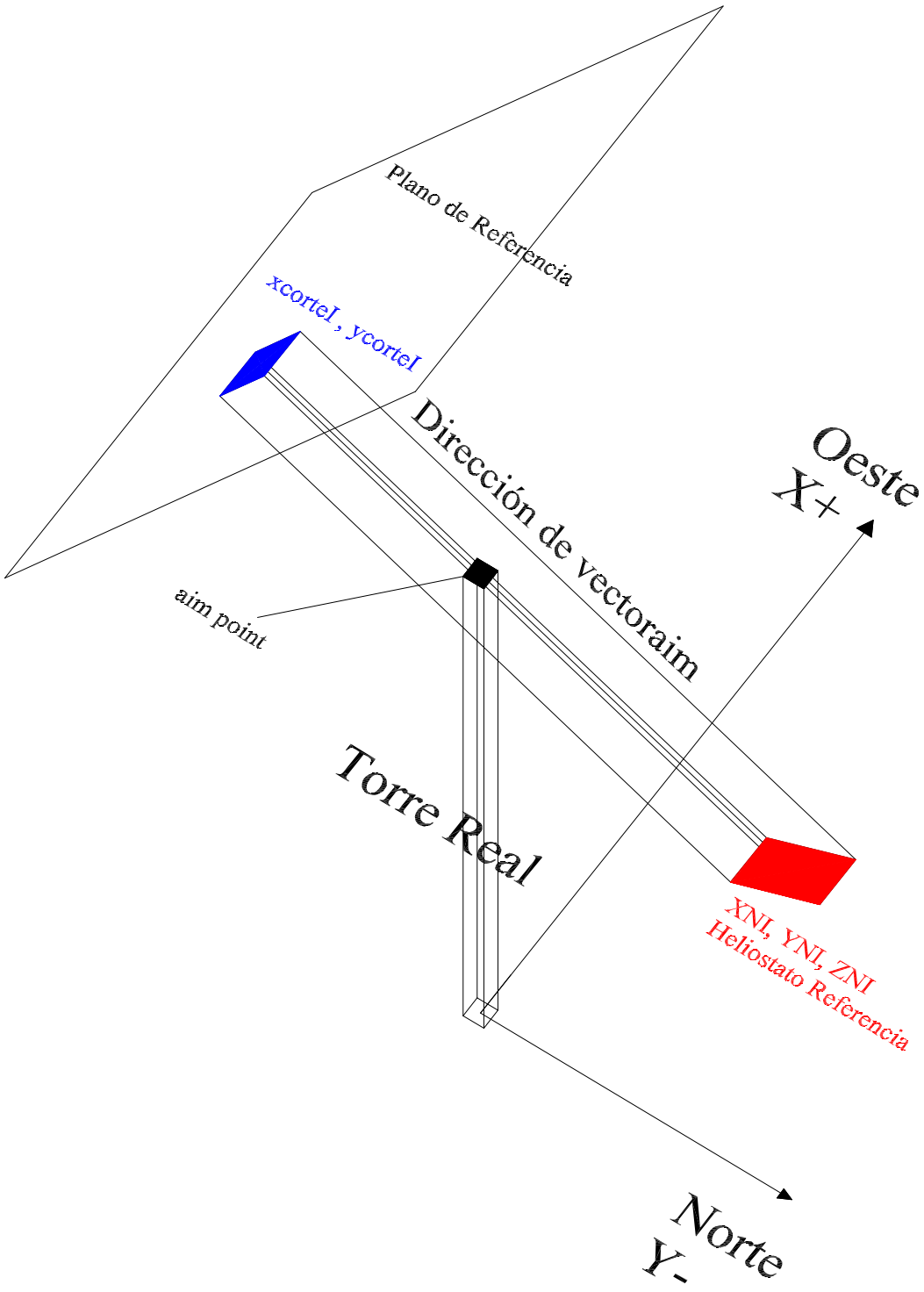


Figura 3.7: Procedimiento de cálculo de bloqueos. Plano de referencia en el que se proyectan lo helióstatos de un grupo de vecinos. En este caso existe un plano de proyección distinto para cada grupo, puesto que debe ser perpendicular a la dirección que une el centro del helióstato con el aim point, y esta dirección cambia para cada uno de ellos.

Función bloqueosesfera

Función llamada por **optimiza3D** y por **performance3D** para calcular las pérdidas por bloqueos en el punto de diseño y las pérdidas medias anuales respectivamente.

```
function [BLOQUEOSTOTAL]=bloqueosesfera(Aimingpoint,XN,YN,ZN,parejasvecinos)
```

argumentos:

Aimingpoint	*vector con las coordenadas del receptor (1x3)*
XN, YN, ZN	*matrices con las coordenadas x,y,z respectivamente, de los vértices de los helióstatos en su posición óptima (nº de helióstatos x 4)*
parejasvecinos	*matriz que indica los' n' helióstatos más próximos a uno dado [(m x n) x 2], donde 'm' es el número total de helióstatos.*

El receptor suele estar situado en la parte más alta de la torre, pero suponer que la coordenada z del objetivo o target es la altura de la torre es una simplificación, ya que normalmente la propia altura del receptor da lugar a un rango para el objetivo, que no es realmente un foco puntual exacto sino una superficie; en esta función, sin embargo se considera que la coordenada z del target es igual a la altura de la torre y que tenemos un foco puntual.

XN, YN, ZN y *parejasvecinos* tienen idéntico significado que para **sombrasplano**.

parámetros de salida:

BLOQUEOSTOTAL	*vector que indica el tanto por uno de bloqueo de cada helióstato (nº de helióstatos x 1)*

Se determinan con este vector las pérdidas por bloqueos en cada espejo.

descripción:

? Ahora se hallan los bloqueos utilizando un solo plano de proyección. Para ello lo que se hace es aproximar las direcciones de proyección, que en la función **bloqueosplano** eran paralelas al *vectoraim* de cada helióstato de referencia, con líneas que van desde cada punto a proyectar hasta el *Aimingpoint*, a modo de radios.

Por otra parte, el plano de proyección será ahora una esfera de radio unidad, alrededor del *Aimingpoint*.

El error cometido será pequeño si se piensa en las grandes distancias que separan los helióstatos del receptor, comparadas con las dimensiones de los espejos.

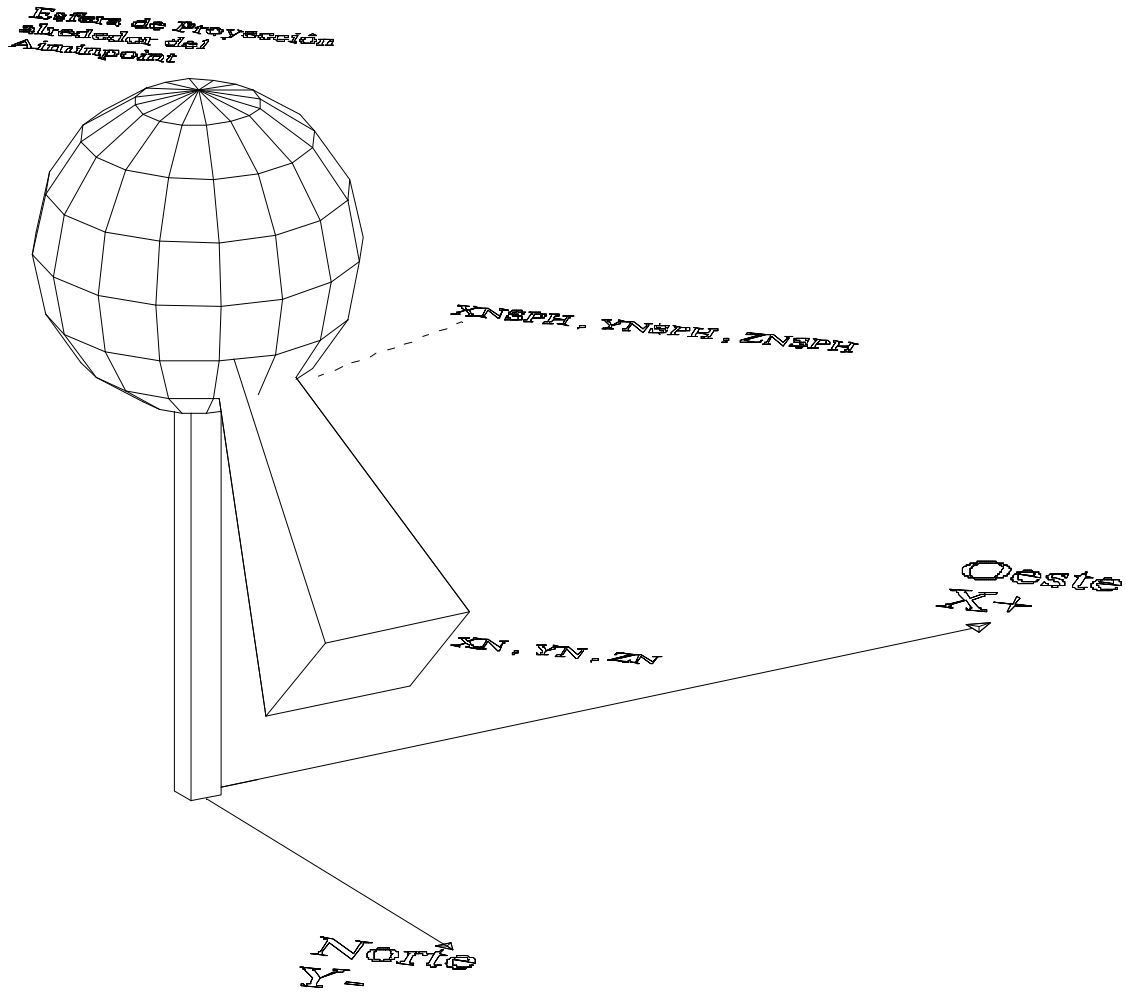


Figura 3.8: Procedimiento simplificado de cálculo de bloqueos. Ahora se proyectan los vértices de todos los helióstatos en una esfera de radio unidad, centrada en el Aim Point. Las direcciones de proyección son rayos que parten de los vértices hasta el aim point.

? Pasando a describir el proceso más detalladamente: en primer lugar, se llama la función **proyesfera** que calcula las proyecciones de los vértices de los helióstatos sobre una esfera de referencia de radio unidad, de centro el *Aimingpoint*. Con las coordenadas de los vértices proyectados, XNSPH, YNSPH, ZNSPH y los vecinos (dados por *parejavecinos)* la función **solapevecinosesf** proporciona el área de los espejos, proyectada en la esfera de referencia, las parejas que se bloquean, y los solapamientos que se producen, que lógicamente representan las pérdidas por bloqueos que se producirán en el helióstato.

? Una vez conocidos los solapamientos en bloqueos de todas las parejas, se determina cual de los helióstatos de cada pareja bloquea al otro; esto se hará viendo cuales son las distancias medias de cada espejo al *Aimingpoint*.

? Por último, calculamos las pérdidas porcentuales en los helióstatos, dividiendo los solapamientos por el área proyectada de cada uno de ellos, dando como resultado la variable *BLOQUEOSTOTAL .*

Función spillage

Función llamada por **optimiza3D** y por **performance3D** para calcular las pérdidas por spillage, en el punto de diseño y las medias anuales, respectivamente.

```
function[SPILLGEO]=spillage(VectorAim,distanAim,VectorRec,Rrec,Aim,SIGMA)
```

argumentos:

VectorAim	*matriz de los cosenos directores de los rayos que unen los helióstatos con el aim point (nº de helióstatos x 3)*
distanAim	*distancia del centro de cada helióstato del campo al aim point (nº de helióstatos x 1, metros) .*
VectorRec	*vector unitario perpendicular al plano del receptor; su sentido es hacia el campo de helióstatos (1x3).*
Rrec	*radio de la apertura del recetor (metros)*
Aim	*vector con las coordenadas del receptor (1x3)*
SIGMA	*desviación estándar*

Aim y *distanAim* ya han sido usados en las funciones de bloqueos y atenuación respectivamente, con idéntico significado lógicamente. Rrec y SIGMA son datos introducidos por el usuario, pasados como argumentos por la función **optimiza3D**.

parámetros de salida:

SPILLGEO	*vector de pérdidas de cada helióstato por spillage geométrico, en tanto por uno.*

SPILLGEO es un vector, también de dimensiones *nº de helióstatos x 1*, porque obtenemos un valor de esta pérdida para cada helióstato.

descripción:

? Las pérdidas por ‘spillage’ en cada helióstato ocurren cuando un rayo de sol reflejado por el área de un espejo, no siendo ya interceptado en su camino al receptor, no es captado por éste, debido principalmente a dos razones:

1) a causa de imperfecciones especulares en la superficie de los helióstatos, la dirección de reflexión del rayo, no coincide con la teórica.

2) el tamaño finito del receptor, sobre todo teniendo en cuenta la inclinación con la que se disponga, hace que algunos rayos escapen a su alcance.

? Para modelar ambos fenómenos en cada helióstato, se ha utilizado una función de distribución de probabilidad de tipo Normal ó Gaussiana bidimensional e independiente, en ***x*** e ***y*** :

$$f(x,y) = \frac{1}{s_1 \cdot \sqrt{2p}} \cdot e^{-\frac{(x-m_1)^2}{2s_1^2}} \cdot \frac{1}{s_2 \cdot \sqrt{2p}} \cdot e^{-\frac{(y-m_2)^2}{2s_2^2}} = f_1(x) \cdot f_2(y)$$

donde : s_1 es la desviación estándar en el **eje x** de integración en el área del receptor

s_2 es la desviación estándar en el **eje y** de integración en el área del receptor

$m_1 = m_2 = 0$ son las medias correspondientes

? En la función se aplicarán estos conceptos de la siguiente forma:

$$prob = \int_{-R_{receptor}}^{R_{receptor}} \int_{-R_{receptor}}^{R_{receptor}} f_1(x) \cdot f_2(y) \cdot dx \cdot dy$$

donde: $s_1 = s_2 = s$

$$s = \sqrt{s_x^2 + s_y^2}$$

$$s_x = \frac{SIGMA \cdot distanaim}{1000}$$

$$s_y = \frac{SIGMA \cdot distanaim}{cosangulo \cdot 1000}$$ ⇨ se ve como aquí se ha tenido en cuenta el hecho de que la inclinación del receptor disminuye su área útil en el **eje y** (no así en el **eje x**, ya que se supone que éste es el eje de giro) respecto a la eficacia de captación de los rayos provenientes de los helióstatos.

cosangulo representa el coseno del ángulo formado entre la perpendicular al plano del receptor*(vectorec)* y la línea que une el centro de éste con el centro del helióstato*(vectoraim).*

SIGMA representa la desviación típica del ángulo que forma el rayo real reflejado respecto al teórico, en un helióstato, expresado en miliradianes

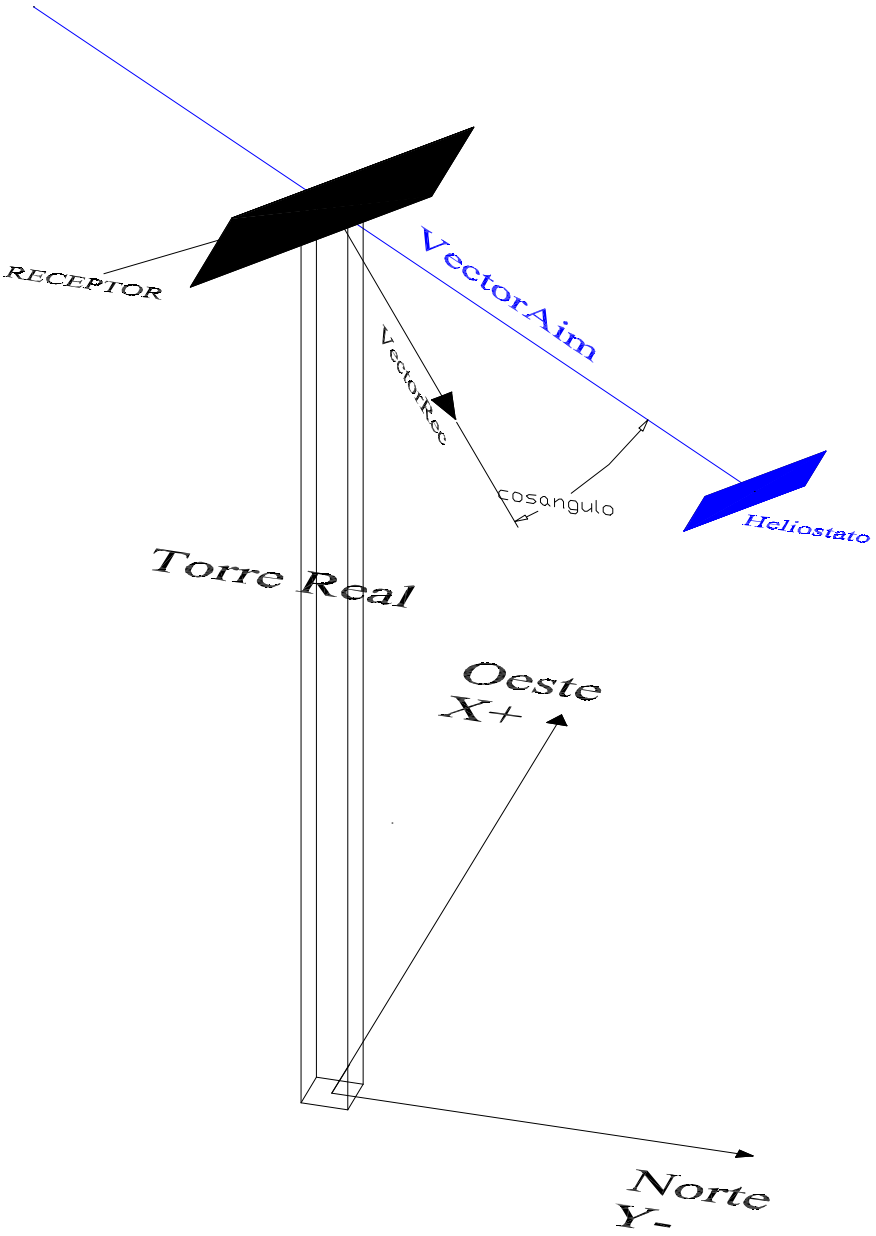


Figura 3.9: Elementos que intervienen en el procedimiento de cálculo de las pérdidas por spillage. Se muestra la posición relativa entre heliosato receptor y torre.

? Por último, las pérdidas por 'spillage' para cada helióstato se modelan como $1 - prob$

Función sombraterreno

Función llamada por **optimiza3D** y por **performance3D** para calcular las pérdidas por sombras provocadas por el terreno en el punto de diseño y las pérdidas medias anuales respectivamente.

```
function[SOMBRASTERRENO]=sombraterreno(PendienteTerreno,cosinc,M)
```

argumentos:

PendienteTerreno	*inclinación del terreno que se esté modelando (grados)*
cosinc	*vector con los cosenos directores de la radiación solar incidente*
M	*número de helióstatos del campo*

PendienteTerreno es el ángulo que forma el suelo de la parcela considerada con respecto a la horizontal; se toma un valor constante. Es un dato de partida para nuestro programa.

Cosinc es un vector de dimensiones (1x3) que modela la dirección de la radiación solar incidente. Es calculado por **solpos** y pasado por parámetros

parámetros de salida:

SOMBRASTERRENO	*vector que indica si el helióstato está o no en sombra (nº de helióstatos x 1)*

Este vector toma valor 1 si el helióstato está en sombra debido al terreno y cero si no lo está; no se han tenido en cuenta sombras parciales.

descripción:

? El cálculo de las sombras se efectúa de la siguiente manera: con el ángulo de la pendiente se calcula la normal al terreno; una vez conocida la normal se evalúa el ángulo que forma con la dirección incidente del sol, mediante el producto escalar de ambos vectores.

Si el valor de este producto es mayor que cero, es que ambos vectores forman un ángulo agudo, el terreno quedará en sombra y se asigna potencia cero a los helióstatos.

Por el contrario, cuando el producto sea negativo, el ángulo formado por la dirección de los rayos será obtuso y el terreno estará libre de sombras.

? Es claro que una zona determinada estará en sombras o no dependiendo de la dirección de incidencia de la luz del sol y variando por tanto para cada día y hora del año. Puede ocurrir por tanto que alguna de las horas de los días en los que se evalúen las medias anuales, den potencia nula para el campo.

También es obvio que está situación se presentará y será más desfavorable cuando el terreno tenga pendiente negativa, como podemos observar en la siguiente figura:

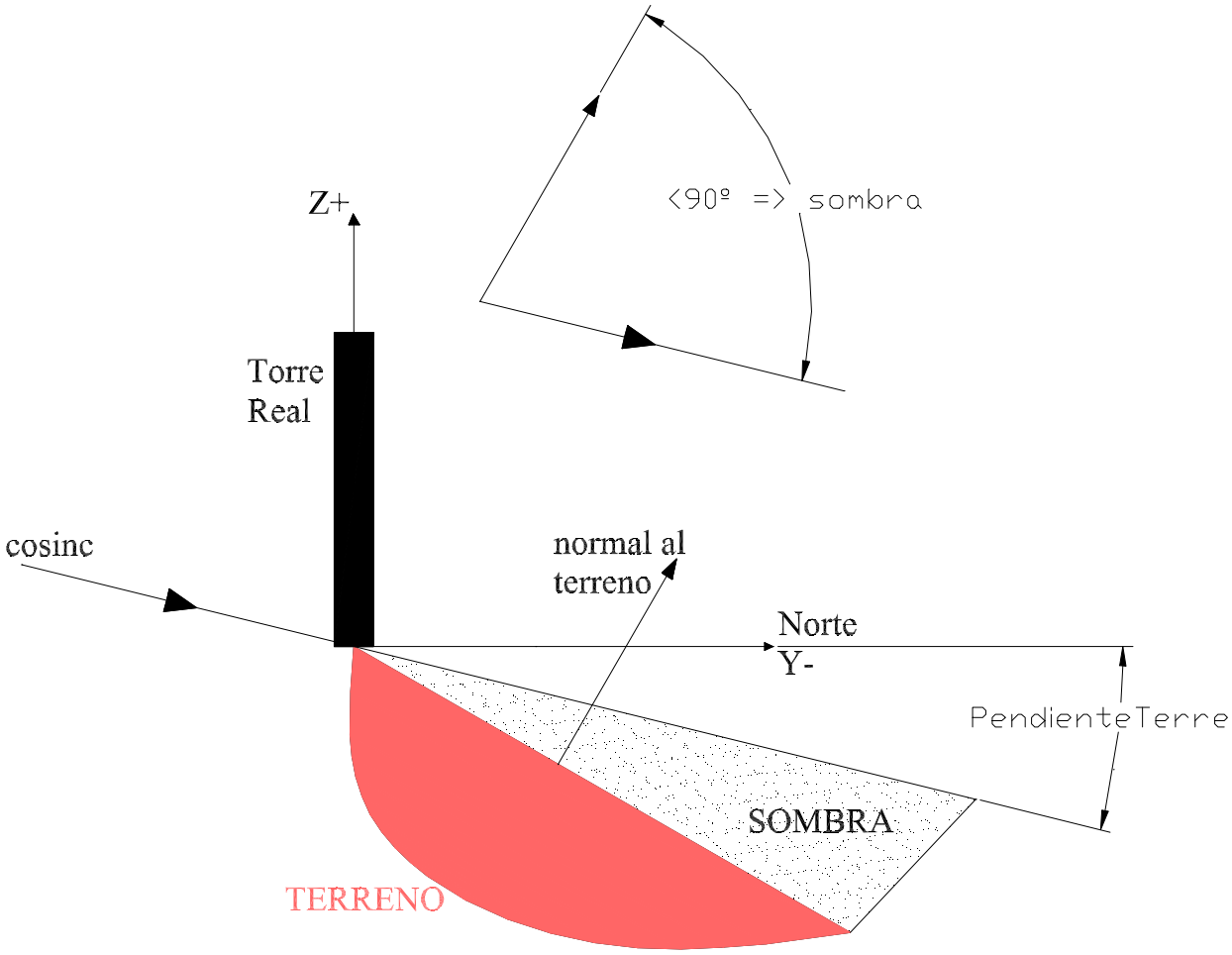


Figura 3.10: Procedimiento de cálculo para las sombras del terreno. Se muestran en la figura la dirección solar incidente y el ángulo que forma con la normal al terreno, para determinar así que parte de este queda en sombra (parte punteada en la figura).

? Como se desprende de esta explicación, este método es totalmente válido cuando el terreno sea una superficie plana. Si no es así, se tendría que conocer el valor del ángulo del terreno con la horizontal en cada punto. En la función **malla 3D** se ha explicado que los terrenos inclinados se modelan como una superficie cónica o como un plano, siendo, pues esta función exacta para este último caso. Señalar que el caso del cono es especialmente complicado, porque también habría que tener en cuenta las sombras que se puede dar a si mismo.

Función minpendiente

Función llamada por **optimiza 3D** que calcula la pendiente límite para que no aparezcan sombras en el terreno.

```
function[minPendienteTerreno]=minpendiente(Latitude,Ndays,Nhours,TILTPLANE)
```

argumentos:

Latitude	*coordenada geográfica (grados)*
Ndays, Nhours	*vectores con los días y las horas donde se evalúa el performance*
TILTPLANE	*inclinación del terreno (grados)*

Todos los argumentos usados en esta función son datos pasados por parámetros por **optimiza 3D**

parámetros de salida:

minPendienteTerreno	*valor de la pendiente (grados)*

El resultado de esta función no es utilizado en ningún cálculo del programa, simplemente permite conocer el valor de la pendiente del terreno para el cual, en ningún momento del día llegaría insolación a los helióstatos.

descripción:

Calcula las direcciones incidentes del sol proyectadas sobre el plano YZ, para cada día y hora utilizados para hallar las medias anuales. El mínimo ángulo formado entre esas proyecciones y el eje Y, sería la pendiente del terreno límite para evitar las sombras del terreno sobre el campo de helióstatos. **(ver Figura 3.10)**

Evidentemente esto solo sería válido para un terreno en forma de pendiente plana; para un terreno en forma de cono sería necesario tener en cuenta otros factores más complejos.

Función performance3D

Función llamada por **optimiza3D** para calcular la potencia y las pérdidas medias anuales, de cada helióstato y la potencia media anual total del campo.

```
function [MedhPOT,MedhCOSEFF,MedhSOMB,MedhBLOCK,MedhATEN,MedhSpill]=
performance3D(Latitud,Ndays,Nhours,RADIACION,Aim,Campo,THT,RTOWER,
ANCHURAH, ALTURAH, PEDEST, TILTREC, Rrec, SIGMA,TILTPLANE);
```

argumentos:

Latitud	*coordenada geográfica (grados)*
Ndays	*vector con los días del año donde se evalúa el performance*
Nhours	*vector con las horas de los días anteriores en las que se evalúa el performance*
RADIACION	*densidad de radiación solar en la zona* (kW/m^2)
Aim	*vector con las coordenadas del receptor (1x3)*
Campo	*matriz con las coordenadas x, y, z del campo colector (nº de helióstatos x3)*
THT	*altura de la torre (metros)*
RTOWER	*radio de la torre (metros)*
ANCHURAH, ALTURAH, PEDEST	*dimensiones de los helióstatos y altura de su pedestal (metros)*
TILTREC	*inclinación del receptor respecto a la vertical (grados)*
Rrec	*radio de la apertura del receptor (metros)*
SIGMA	*desviación estándar*
TILTPLANE	*inclinación del terreno (grados)*

Todos estos argumentos son datos pasados por parámetro desde **optimiza3D,** salvo *Campo* que ha sido calculado previamente por **malla3D .**

La elección de los días y las horas utilizados para la extrapolación de los cálculos del punto de diseño a los cálculos anuales, se ha basado en un criterio de representatividad de las distintas posiciones del sol a lo largo del año. [1]

parámetros de salida:

MedhPOT *potencia media anual de cada helióstato del campo de prueba (kilowatios)*

MedhCOSEFF, MedhSOMB, MedhBLOCK, MedhATEN, MedhSpill
pérdidas medias anuales por efecto coseno, sombras, bloqueos atenuación y spillage geómetrico (kilowatios)

descripción:

? En el primer grupo de sentencias se inicializarán las variables con las que va a trabajar la función, reservando memoria para almacenar la potencia y las pérdidas acumuladas en el año para cada helióstato del campo y también un contador, *CONTADORHORAS* en el que se reflejará el número de horas reales en las que se evalúan dichas pérdidas, que no son todas las del año. Es suficiente aproximación al cálculo anual el tomar un número de días representativos y dentro de éstos, un número de horas. El excesivo tiempo de cálculo que esto supondría no se ve compensado con unos resultados mucho más aproximados.
La función **vecinos3D** proporciona los 10 vecinos más próximos a cada helióstato.

? En el segundo grupo de sentencias se ejecuta exactamente el mismo proceso seguido en **optimiza 3D** para calcular las pérdidas de cada espejo, primero en tanto por uno y después en kW .
En **optimiza 3D**, este proceso se realizaba solo para el cálculo en el punto de diseño, es decir, para un único día del año y una única hora de ese día. Ahora, **performance 3D** efectúa los cálculos para los cinco días del año dados por *Ndays* y para las cuatro horas del día dadas por *Nhours,* lo que supone un total de veinte momentos distintos del año, en los que cada helióstato del campo dado tendrá distintas pérdidas y proporcionará distintas potencias que se acumulan en la variables inicializadas a este fin; también se actualiza el *CONTADORHORAS.*

? Por último, en el tercer grupo de sentencias se obtienen los valores medios anuales por helióstato, de potencia, *MedhPOT* y pérdidas, *MedhCOSEFF, MedhSOMB, MedhBLOCK*, *MedhATEN, MedhSpill*, en kW, simplemente dividiendo los valores acumulados por el valor de la variable *CONTADORHORAS.* Para obtener los valores de energía media anual, basta multiplicar los resultados en kW por el número estimado de horas útiles al año (unas 2800).
También se proporciona el valor total de potencia media anual del campo objeto de cálculo, *TOTALPOT,* resultado de sumar la de cada uno de los espejos.

? Para una visión más clara del programa se adjunta a continuación un diagrama de flujo del mismo, que también se realizará en otras funciones, cuando el código sea más complicado. También puede ayudar a la comprensión del funcionamiento del programa, el código del mismo que aparece como anexo al final del trabajo.

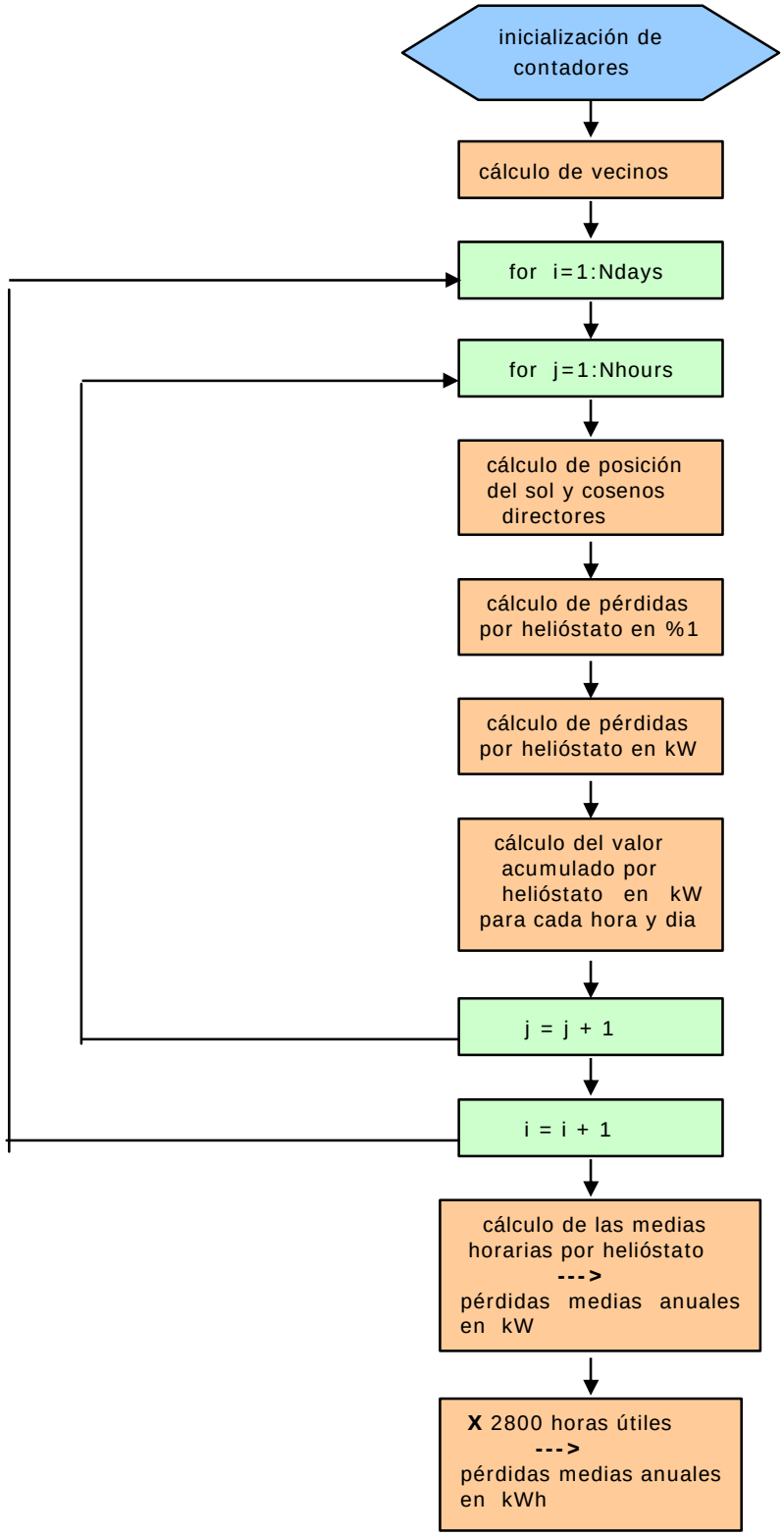


Diagrama de flujo performance 3D

Función solpos

Esta función, llamada por **optimiza3D** primero y posteriormente por **performance3D**, calcula los ángulos que forman los rayos solares incidentes en la superficie terrestre, en función de una localización geográfica determinada en dicha superficie.

```
function [AZIMUT,ELEVATION]= solpos(Latitude,N,Hour,TILT)
```

argumentos:

Latitude	*coordenada geográfica (grados)*
N	*día elegido para calcular la posición solar*
Hour	*hora elegida para el cálculo (hora solar (10h 30'))*
TILT	*ángulo de la superficie inclinada (grados)*

Estos argumentos son datos fijados por el usuario, pues dependen de la localización de la planta; son los que determinan dicha posición, de forma que a partir de ellos, se puede calcular la posición del sol para un día y una hora dados.

parámetros de salida:

AZIMUT	*ángulo azimutal, 0 sur, pi/2 oeste (radianes)*
ELEVATION	*ángulo elevación, 0 en el ecuador (radianes)*

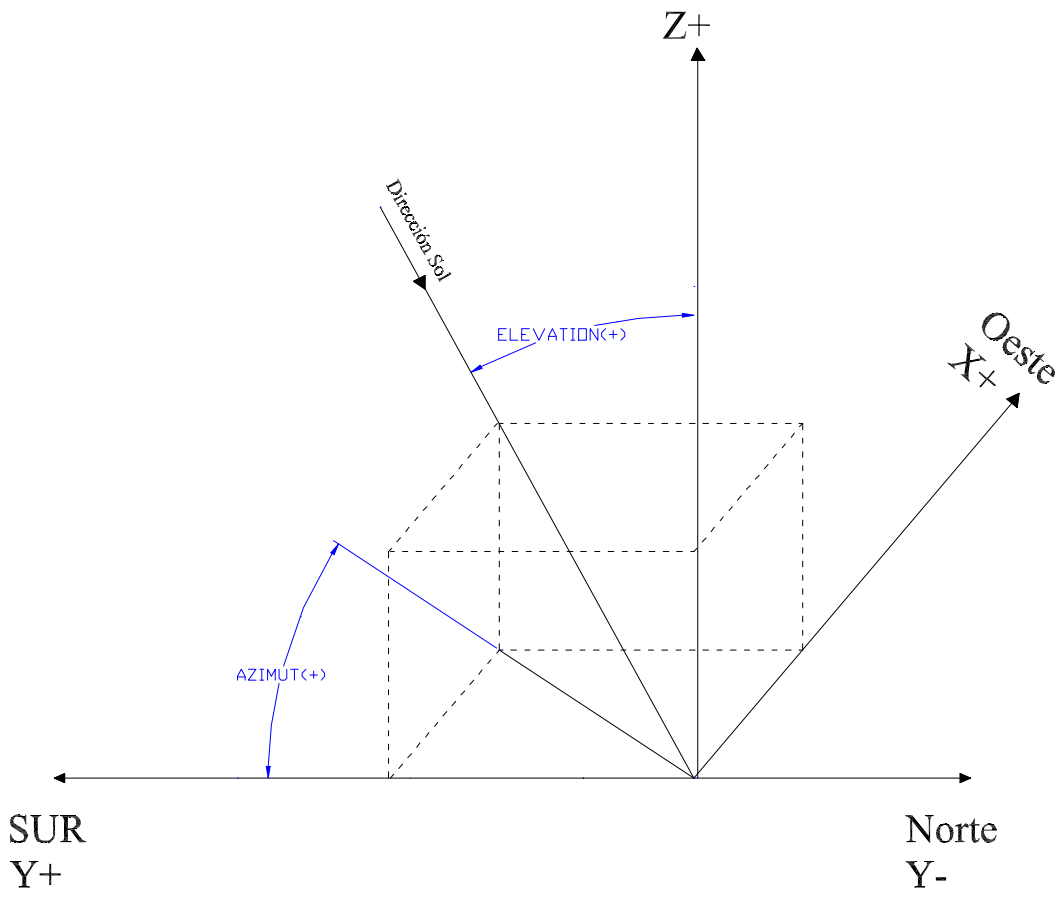
La posición de un rayo de sol genérico, como la de cualquier otro vector, viene determinada conociendo dos de los ángulos que forme el vector con los ejes del sistema de coordenadas al que se quiera referir. Los ángulos AZIMUT y ELEVATION que se definen para el sistema que se muestra en la figura de la página siguiente, fijan por tanto la posición del sol.

descripción:

? El vector que da la posición del sol especificada por AZIMUT y ELEVATION puede ser calculado, si se ignoran los efectos de refracción en la atmósfera por las siguientes fórmulas: [10]

$$ELEVATIONZ = a\cos\left[\sin(Dec)\cdot\sin(Latitude)+\cos(Dec)\cdot\cos(Latitude)\cdot\cos(W)\right]$$

$$AZIMUT = a\sin\left\{\frac{\sin(W)\cdot\cos(Dec)}{\sin(Dec)}\right\}$$



Función 3.11: Definición de los ángulos azimutal y elevación respecto al sistema de coordenadas utilizado.

W es el ángulo horario medido desde el mediodía:

$$W = (Hour - 12) \cdot 15 \cdot \left(\frac{p}{180} \right)$$

Dec es la declinación terrestre en *radianes* calculada con la siguiente fórmula:

$$Dec = 23.45 \cdot \left(\frac{p}{180} \right) \cdot \sin\left[\left(360 \cdot \frac{(284 + N)}{365} \right) \cdot \left(\frac{p}{180} \right) \right]$$

? Estas fórmulas han sido desarrolladas para calcular los ángulos que determinan la posición del sol tomando la superficie terrestre como un plano horizontal. Para tener en cuenta la inclinación del terreno se han modificado las fórmulas anteriores introduciendo el ángulo que formaría el terreno con la horizontal a la superficie terrestre.

$$ELEVATION = \sin(Dec)\cdot\sin(Latitude)\cdot\cos(TILTR) - \sin(Dec)\cdot\cos(Latitude)\cdot\sin(TILTR) +$$
$$+\cos(Dec)\cdot\cos(Latitude)\cdot\cos(TILTR)\cdot\cos(W) + \cos(Dec)\cdot\sin(Latitude)\cdot\sin(TILTR)\cdot\cos(W)$$

$$AZIMUT = a\cos\left\{\sin(ELEVATION)\cdot\left[\frac{\cos(ELEVATIONZ) - \cos(TILTR)\cdot\cos(ELEVATION)}{(-1)\cdot\sin(TILTR)}\right]\right\}$$

? En **solpos** se aplican estas últimas fórmulas, previos cálculos de los parámetros ***Dec*** y ***W*** y se obtienen sin más AZIMUT y ELEVATION que nos dan la dirección de los rayos de sol en un punto de la tierra, a una hora de un día determinado.

Función cosdirinc

Función que calcula los cosenos directores de los rayos solares incidentes a partir de su AZIMUT y su ELEVATION. Es llamada por **optimiza3D** y por **performance3D**

```
function [cosinc]= cosdirinc(AZIMUT, ELEVATION)
```

argumentos:

AZIMUT	*ángulo azimutal, 0 sur, pi/2 oeste (radianes)*
ELEVATION	*ángulo elevación, 0 en el ecuador (radianes)*

Estos ángulos que sitúan al rayo de sol que incide en el helióstato, en el sistema de coordenadas elegido, han sido calculados por **solpos** para un punto terrestre, un día y una hora dados.

parámetros de salida:

cosinc	*vector con los cosenos directores de una dirección(1x3)*

Esta es la dirección de los rayos del sol dada por el valor de los ángulos AZIMUT Y ELEVATION.

descripción:

? La función realiza un cambio de sistema de coordenadas: de unas coordenadas angulares tipo esféricas pasa a cosenos directores.

Aunque en el uso particular que se hace de **cosdirinc**, el vector dado por AZIMUT, ELEVATION es un rayo de sol, esta función puede obtener los cosenos directores de cualquier otro vector, siempre que venga determinado por unas coordenadas angulares iguales a las anteriores.

? Así la transformación viene dada por:

$$\cos g = -\cos(ELEVATION)$$

$$\cos b = -\sin(ELEVATION)\cdot\cos(AZIMUT)$$

$$\cos a = \pm\sqrt{1-(\cos b)^2-(\cos g)^2}$$

? En la **Figura 3.12** se muestra con mayor claridad la relación entre las distintas variables:

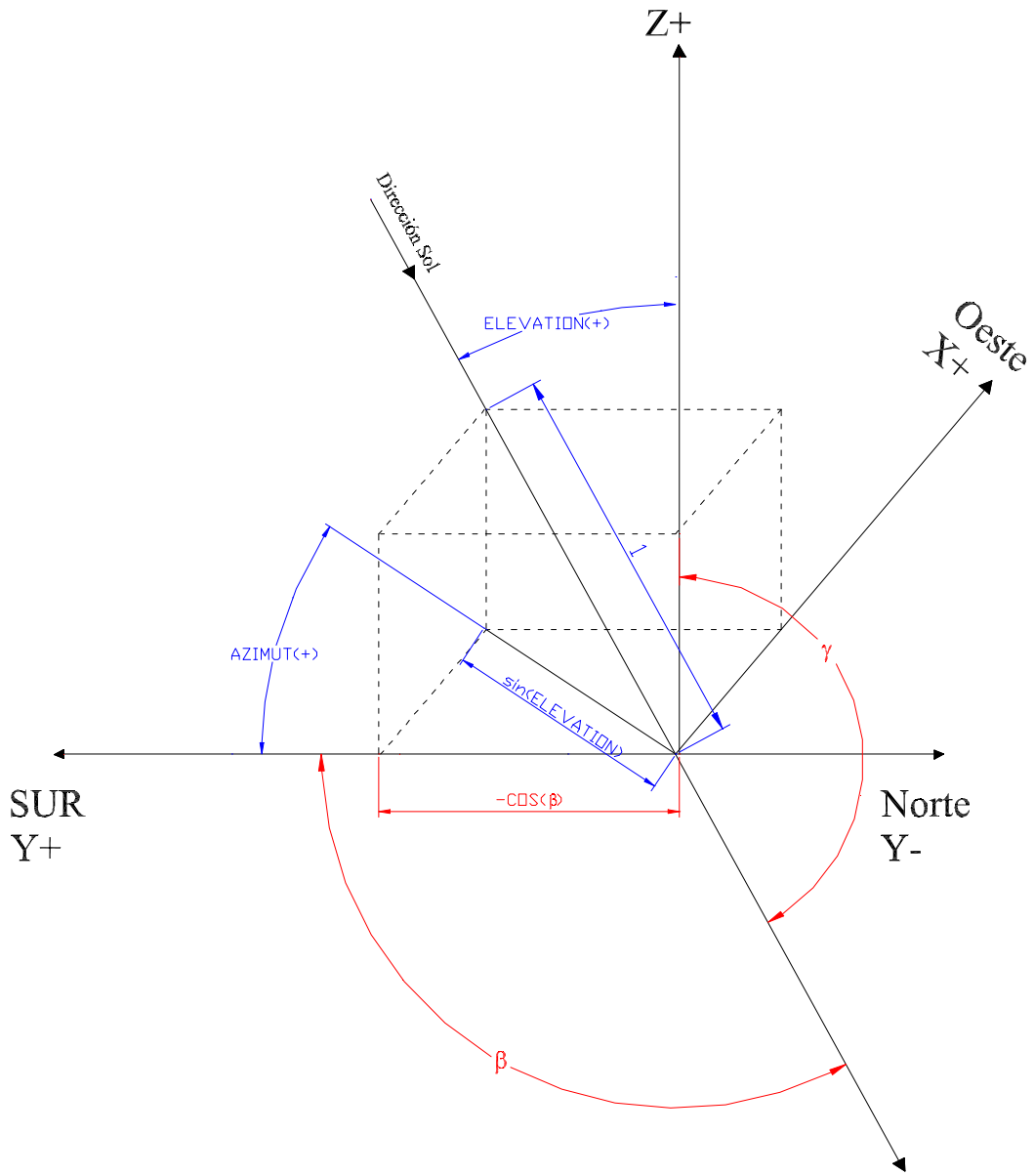


Figura 3.12: Relación entre las coordenadas tipo esféricas r, azimutal, y elevación, y las cooredenadas a, ß, g que representan los cosenos directores. El cambio de un sistema a otro es necesario porque hay cálculos que se realizan más fácilmente con la dirección expresada en función de los cosenos directores y otros cuando se expresa como azimut y elevación.

Función azimelev

Función llamada por **vértices** y **solapevecinosesf** que calcula los ángulos de AZIMUT y ELEVATION de un vector a partir de sus cosenos directores.

```
function [AZIMUT, ELEVATION]= azimelev(cosa,cosß,cos?)
```

argumentos:

cosa, cosß, cos? *vector con los cosenos directores de una dirección (1x3)*

cosa, cosß, cos? son los cosenos directores de los rayos solares cuando **azimelev** es usada por **solapevecinosesf**, y es igual a los de la dirección normal al espejo cuando es usada por **vértices.**

parámetros de salida:

AZIMUT *ángulo azimutal, 0 en dirección Sur, 90º en dirección oeste (radianes)*

ELEVATION *ángulo elevación, 0 en el ecuador (radianes)*

Coordenadas angulares tipo esféricas con las que se sitúan distintas direcciones.

descripción:

? Esta función realiza la tarea inversa a **cosdirinc. Ver Figura 3.12.** Es decir, realiza el cambio inverso de coordenadas.

? Así la transformación viene dada por:

$$ELEVATION = a\cos(\cos g)$$

$$AZIMUT = a\cos\left(\frac{\cos b}{\sin(a\cos(\cos g))}\right)$$

Función vecinos 3D

Función llamada por **optimiza3D** y por **performance3D**. Su misión es encontrar los 'n' helióstatos más próximos entre sí o vecinos en el campo.

```
function [parejas]=vecinos3D(XCAMPO,YCAMPO,ZCAMPO,n)
```

argumentos:

XCAMPO, YCAMPO, ZCAMPO	*vectores con las respectivas coordenadas x, y, z de los pedestales de los espejos (nº de helióstatos x 1)*
'n'	*número de vecinos que se calculan*

XCAMPO, YCAMPO, ZCAMPO son calculados por la función **malla3D**, son las columnas de la matriz *campo* que proporciona las posiciones de los helióstatos.

'n' es un argumento variable, cuyo valor es fijado por el usuario; en función de su valor se calculan más o menos vecinos.

parámetros de salida:

parejas	*proporciona las parejas de vecinos; sus dimensiones dependen del tamaño del campo y del número de vecinos que se quiera calcular. [(mxn)x2] .' m' es el número de helióstatos.*

parejas es una matriz en la que la primera columna indica el helióstato del que se están calculando los vecinos y la segunda indica el helióstato vecino a él. Cada helióstato aparecerá 'n' veces, ya que tiene 'n' vecinos.

descripción:

? La estrategia para conseguir localizar los helióstatos más próximos a uno dado es la siguiente:

Se toman las primeras coordenadas *x, y, z* almacenadas en *campo,* que se corresponden con el primer espejo que se dispondría, y se calcula la distancia entre éste y todos los demás espejos del campo.

Se ordenan estas distancias de menor a mayor y se le asigna el índice del helióstato al que le corresponde esa distancia, es decir, se conoce cuales son los más cercanos al primero.

Se construye la matriz de *parejas,* que en la primera columna indicará el helióstato del que se están calculando los vecinos y en la segunda, el índice que identifica al vecino, el más próximo será el primero y se relacionarán tantos como indique *'n'.* Cuando se haya contabilizado una pareja de vecinos que luego vuelva a aparecer, por ejemplo la (1,3) y la (3,1), esta última ya no se tiene en cuenta.

Por último, indicar que este procedimiento se realiza para todos los helióstatos del campo, obteniéndose la matriz final de *parejas,* que permite conocer cuales son los helióstatos más próximos a otro cualquiera del campo, necesario para conocer las sombras y los bloqueos.

Función vertices

Función llamada por **optimiza3D** y por **performance3D** para calcular los vértices de los helióstatos en sus posiciones óptimas.

```
function [XN, YN, ZN, Normalhel, Cosref]= vertices(pedest,alto,ancho,
Cosinc,Aim)
```

argumentos:

pedest	*matriz con las coordenadas del campo colector (nº de helióstatos x 3)*
alto, ancho	*dimensiones de los espejos (metros)*
Cosinc	*vector con los cosenos directores de los rayos incidentes sobre los helióstatos (1 x 3)*
Aim	*vector con las coordenadas del receptor (Mx3). 'M' es el nº de helióstatos.*

pedest ha sido calculado por **malla3D** y *Cosinc* por **cosdirinc**, mientras que *alto, ancho y Aim* son datos. Los elementos de la matriz *Aim* son las coordenadas, *x, y, z* del receptor repetidas el nº de helióstatos de veces.

parámetros de salida:

XN, YN, ZN	*matrices con las coordenadas x, y z de los vértices de los helióstatos en su posición óptima (nº de helióstatos x nº de vértices)*
Normalhel	*matriz en la que se guarda la dirección normal a la superficie de cada helióstatos para cada posición (nº de helióstatos x3)*
Cosref	*matriz cuyas componentes son los cosenos directores de los rayos reflejados por los helióstatos hacia el receptor (nº de helióstatos x3)*

descripción:

? En primer lugar, hay que aclarar qué se entiende por posición óptima de los vértices de un helióstato. Dada la posición del centro del espejo del helióstato (*pedest)*, es necesario orientar el plano en el cual está contenido dicho espejo para que, según una dirección del rayo de sol incidente (*Cosinc*), el rayo reflejado correspondiente (*Cosref*) incida sobre el receptor. Para conseguir esto, la perpendicular al plano del espejo deberá coincidir con la bisectriz del ángulo formado por *Cosinc* y *Cosref.*

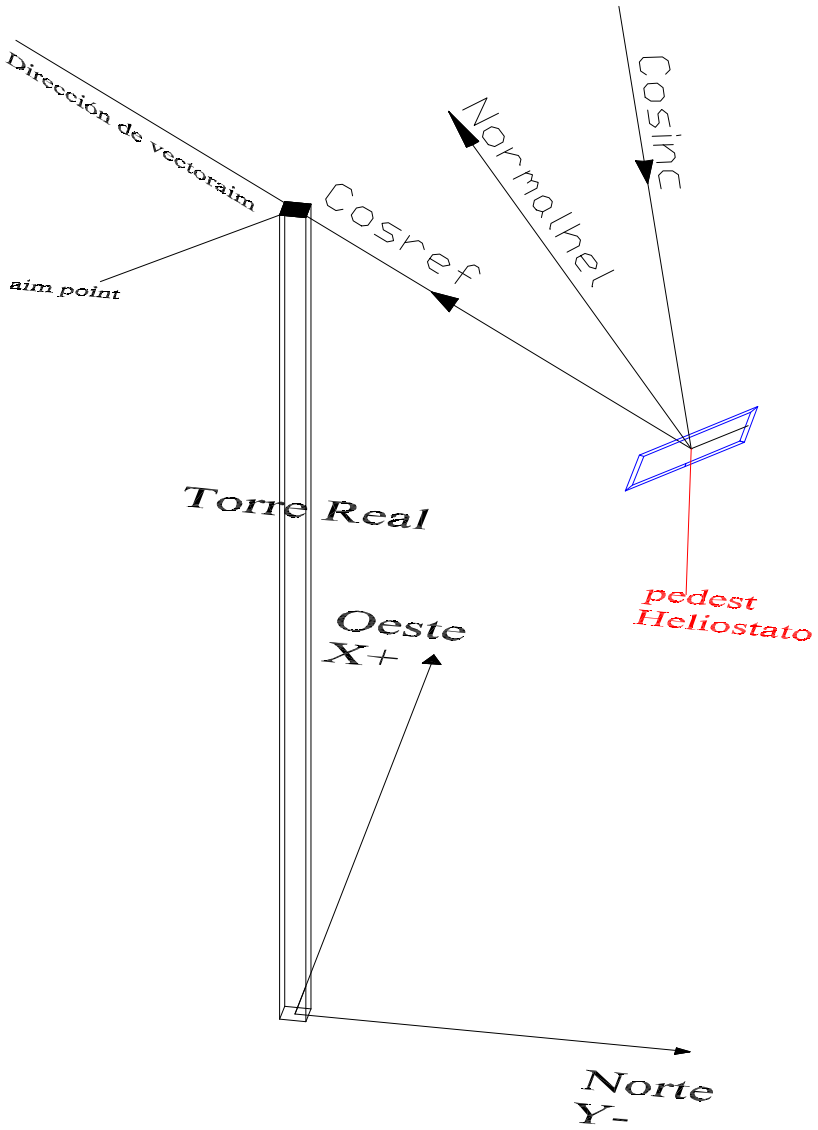


Figura 3.13: Direcciones de los rayos solares incidente y reflejado, y normal al helióstato. Conociendo el rayo incidente, el centro del helióstato y la posición del receptor, queda determinado el plano en el que debe estar contenido el helióstato: perpendicular a la bisectriz del ángulo formado por los rayos incidente y reflejado.

? Aun así, todavía queda un tercer grado de libertad del espejo, que sería la posición del helióstato dentro de este plano, mediante el giro del helióstato alrededor de *Normalhel*, una vez contenido en el plano. En este programa se ha decidido colocar el eje correspondiente a la mayor dimensión del helióstato (el alto casi siempre) contenido en el plano que forman la torre y el punto central del helióstato. De esta forma, y puesto que el mallado se realiza en anillos alrededor de dicha torre, se alcanzará la máxima compactación y densidad de helióstatos posible.

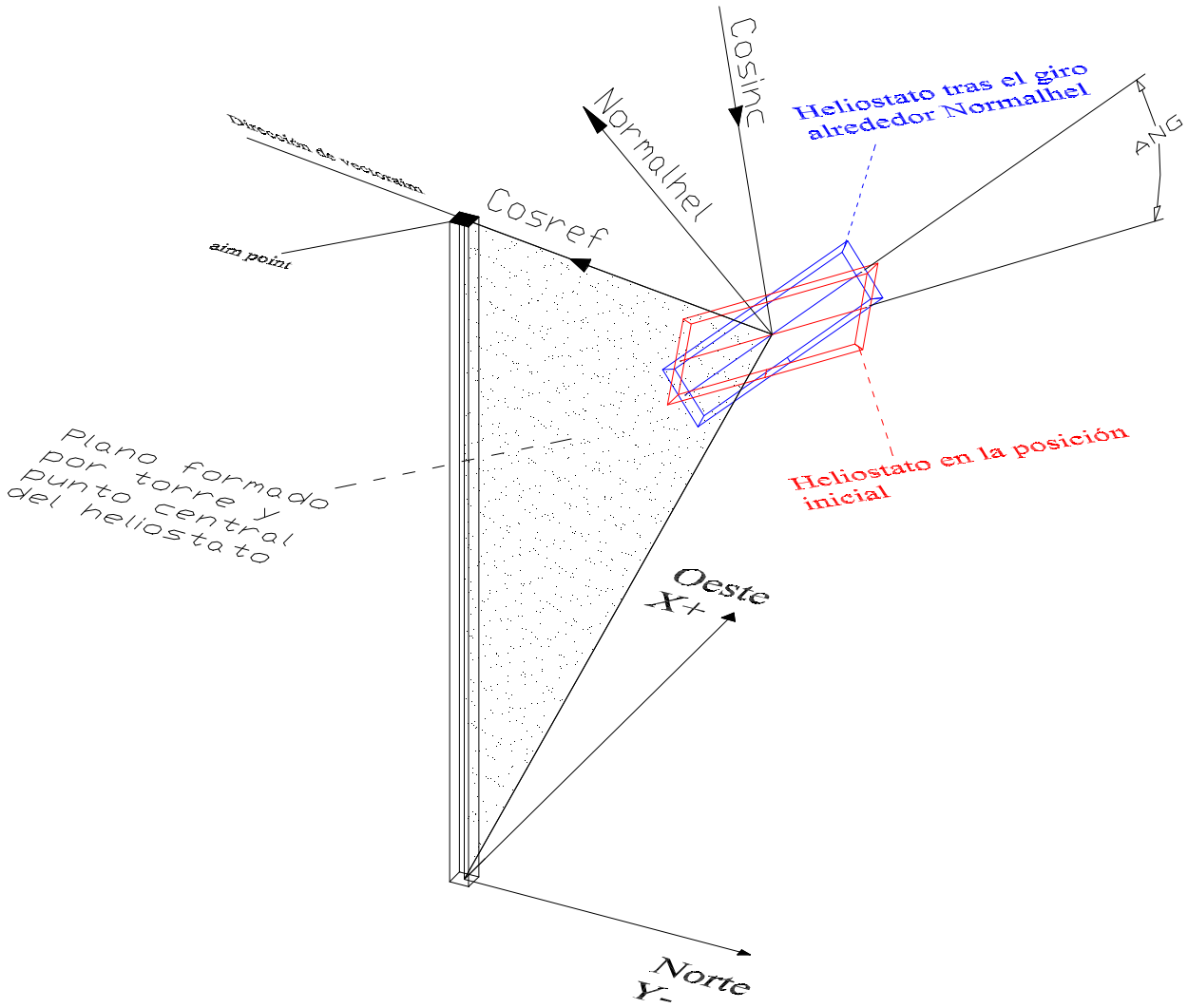


Figura 3.14: Se muestra el helióstato en su posición inicial, (en rojo), antes del giro entorno a normalhel, el plano definido por la torre y el centro del helióstato, y este último en su posición, tras el giro que le da la función giromatriz3, de manera que su eje mayor queda contenido en el plano torre centro heliostato.

? La función **anguloreflex** calcula *Cosref*, es decir, el vector unitario que representa la dirección del rayo reflejado por el espejo hacia el receptor.

? La función **normalinreflex** calcula *Normalhel*, es decir, el vector unitario que representa la dirección normal al plano del helióstato.

? Con la función **azimelev** obtenemos las coordenadas esféricas (como se explicó en el apartado correspondiente a dicha función) de *Normalhel,* utilizadas por la función **giromatriz2** para girar el helióstato.

? A continuación, se gira cada helióstato mediante la función **giromatriz2** desde una posición de referencia dada por la matriz *coordrel* **(mirar Figura 3.15)** hasta hacer que los helióstatos queden contenidos en el plano *Normalhel,* para poder hallar el ángulo que es necesario girar cada helióstato dentro de su propio plano (según se expuso en apartado anterior), dado por el vector ANG.

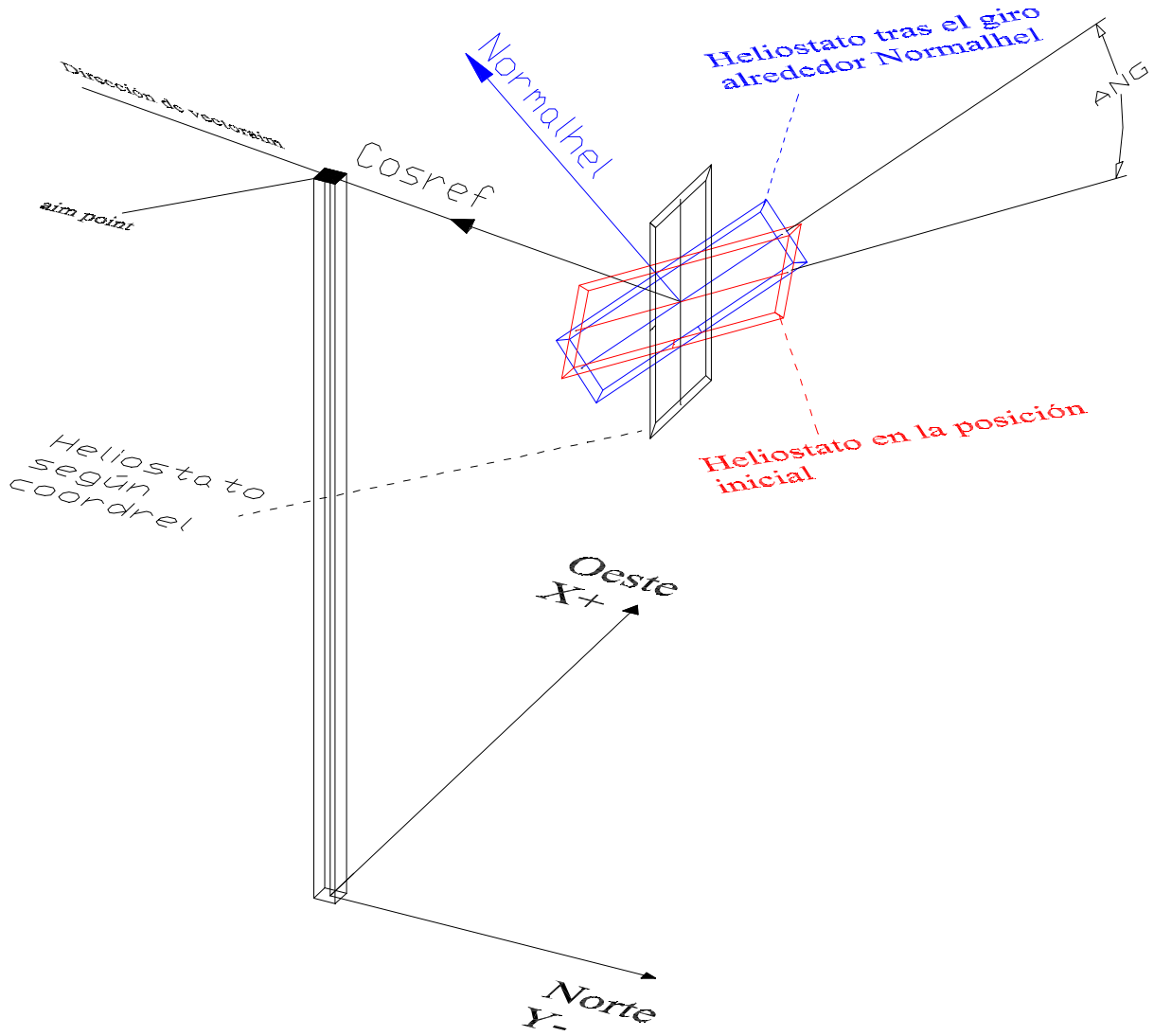


Figura 3.15: Aparecen aquí el helióstato en su posición de referencia (en negro); de esta posición ficticia se pasa a la que se ha denominado inicial (en rojo) que está contenido en el plano perpendicular a normalhel mediante dos giros con la función giromatriz2, y, en azul se representa la posición final que aporta la máxima posibilidad de compactación.

$$coordrel = \begin{pmatrix} -ancho/2 & +ancho/2 & +ancho/2 & -ancho/2 \\ 0 & 0 & 0 & 0 \\ +alto/2 & +alto/2 & -alto/2 & -alto/2 \end{pmatrix}$$

$$MATRIX1=\begin{pmatrix} +\cos(AZIMUT) & -sin(AZIMUT) & 0 \\ +sin(AZIMUT) & +\cos(AZIMUT) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$MATRIX2=\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(ELEVATIONA) & -sin(ELEVATIONA) \\ 0 & sin(ELEVATIONA) & \cos(ELEVATIONA) \end{pmatrix}$$

matrixr = MATRIX1xMATRIX2

? Por último, se usa **giromatriz3** para calcular la matriz que permite realizar los tres giros del helióstato según los tres correspondientes ángulos (AZIMUT, ELEVATION y ANG) y que dejaría ya los helióstatos en su posición final:

$$MATRIX1=\begin{pmatrix} +\cos(AZIMUT) & -sin(AZIMUT) & 0 \\ +sin(AZIMUT) & +\cos(AZIMUT) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$MATRIX2=\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(ELEVATIONA) & -sin(ELEVATIONA) \\ 0 & sin(ELEVATIONA) & \cos(ELEVATIONA) \end{pmatrix}$$

$$MATRIX3=\begin{pmatrix} +\cos(ANG) & 0 & -sin(ANG) \\ 0 & 1 & 1 \\ +sin(ANG) & 0 & +\cos(ANG) \end{pmatrix}$$

matrix= MATRIX1 x MATRIX2 x MATRIX3

Función anguloreflex

Función llamada por **vertices** para calcular los cosenos directores de los rayos reflejados por los helióstatos.

```
function [Cosref]= anguloreflex(pedest,Aim)
```

argumentos:

pedest	*matriz con las coordenadas del campo colector (nº de helióstatos x 3)*
Aim	*vector con las coordenadas del receptor (Mx3). 'M' es el nº de espejos.*

Ambos argumentos son pasados por parámetros por **vértices**.

parámetros de salida:

Cosref	*matriz cuyas componentes son los cosenos directores de los rayos reflejados por los helióstatos (nº de helióstatos x3)*

descripción:

? La función anguloreflex calcula los cosenos directores de los rayos reflejados por los helióstatos. La dirección de estos rayos debe ser la que une el receptor con el centro de los helióstatos.

? Como se conocen las coordenadas x, y, z del centro de los espejos y del receptor, se puede obtener la dirección de reflexión restando dichas coordenadas para cada elemento del campo colector. También se puede calcular la distancia entre ambos puntos como el módulo del vector que los une.

? Los cosenos directores, dados por *Cosref* resultan de dividir las direcciones entre los módulos.

Función normalinreflex

Función llamada por **vertices** que proporciona la normal a una superficie, conociendo las direcciones incidente y reflejada en la misma.

```
function[N]=normalinreflex(I,R)
```

argumentos:

I *cosenos directores de los rayos solares incidentes*
(m x 3) *'m' es el nº de helióstatos.*

R *cosenos directores de los rayos reflejados por el helióstato (m x3).*

I es igual a *Cosinc* calculado por **cosdirinc** y *R* es igual a *Cosref* obtenido por **anguloreflex**.

parámetros de salida:

N *cosenos directores de la dirección normal a una superficie (m x 3).*

N es la dirección en función de la cual se dispondrá el plano del helióstato para que sea perpendicular a ella.

descripción:

? Calcular esta dirección *N* es fundamental para posicionar los elementos del campo colector. La dirección de los rayos incidentes está fijada por el sol y una vez situada la torre y los pedestales de los espejos, las direcciones de apunte también.

¿Cómo se dispone entonces el plano del helióstato? Esta posición queda determinada ya, porque para que sea posible el apunte se tiene que cumplir la ley de la reflexión y ésta dice que la superficie de reflexión debe ser perpendicular a la bisectriz de los rayos incidentes y reflejados. 'N' es precisamente esta bisectriz y es la que proporciona el plano que contiene al helióstato para que sea perpendicular a ella

Función giromatriz2

Función llamada por **vertices** que calcula la matriz que permite el giro de los helióstatos desde su orientación inicial de referencia hasta hacer que queden contenidos en el plano necesario correspondiente.

```
function [matrixr]=giromatriz(AZIMUT,ELEVATION)
```

argumentos:

AZIMUT *ángulo azimutal, 0 en dirección sur, 90º en dirección oeste (radianes)*

ELEVATION *ángulo elevación, 0 en el ecuador (radianes)*

Coordenadas angulares tipo esféricas con las que se sitúan distintas direcciones en función de los cosenos directores que se introduzcan en **azimelev**. En este caso, *azimut* y *elevation* posicionan las direcciones normales de los espejos.

parámetros de salida:

matrixr *matriz de giro (3 x 3)*

descripción:

? La función calcula la matriz de giro correspondiente a partir de los dos correspondientes ángulos (AZIMUT y ELEVATION) que nos dejaría cada helióstato contenido en el plano deseado, a falto del último giro según ANG:

$$MATRIX1=\begin{pmatrix} +\cos(AZIMUT) & -sin(AZIMUT) & 0 \\ +sin(AZIMUT) & +\cos(AZIMUT) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$MATRIX2=\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(ELEVATIONA) & -sin(ELEVATIONA) \\ 0 & sin(ELEVATIONA) & \cos(ELEVATIONA) \end{pmatrix}$$

matrixr = MATRIX1xMATRIX2

Función giromatriz3

Función llamada por **vertices** que calcula la matriz que permite el giro de los helióstatos desde su orientación inicial de referencia hasta hacer que queden en su posición final.

```
function [matrixr]=giromatriz(AZIMUT,ELEVATION,ANG)
```

argumentos:

AZIMUT	*ángulo azimutal, 0 en dirección sur, 90º en dirección oeste (radianes)*
ELEVATION	*ángulo elevación, 0 en el ecuador (radianes)*
ANG	*ángulo que es necesario girar cada helióstato dentro de su propio plano*

parámetros de salida:

matrixr	*matriz de giro (3 x 3)*

descripción:

? La función calcula la matriz que nos permite realizar los tres giros del helióstato según los tres correspondientes ángulos (AZIMUT, ELEVATION y ANG) y que dejaría ya los helióstatos en su posición final:

$$MATRIX1=\begin{pmatrix} +\cos(AZIMUT) & -sin(AZIMUT) & 0 \\ +sin(AZIMUT) & +\cos(AZIMUT) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$MATRIX2=\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(ELEVATIONA) & -sin(ELEVATIONA) \\ 0 & sin(ELEVATIONA) & \cos(ELEVATIONA) \end{pmatrix}$$

$$MATRIX3=\begin{pmatrix} +\cos(ANG) & 0 & -\sin(ANG) \\ 0 & 1 & 1 \\ +\sin(ANG) & 0 & +\cos(ANG) \end{pmatrix}$$

matrix= MATRIX1 x MATRIX2 x MATRIX3

Función proyesfera

Función utilizada por **optimiza3D**, **performance3D**, y **bloqueosesfera** para calcular proyecciones sobre una esfera.

```
function[cosalfa,cosbeta,cosganma]=proyesfera(XN,YN,ZN,XCentroesf,YCentroesf,
ZCentroesf)
```

argumentos:

XN, YN, ZN *matrices con las coordenadas x, y, z de los vértices de los helióstatos en su posición óptima (nº de helióstatos x nº de vértices)*

XCentroesf, YCentrosef, ZCentroesf *coordenadas x, y, z del centro de la esfera de proyección.*

XN, YN, ZN son los vértices de los polígonos que se quieren proyectar en la esfera; como se ha señalado varias veces en funciones anteriores son calculados por **vértices** y representan los vértices de los helióstatos.

XCentroesf, YCentrosef, ZCentroesf definen la esfera de proyección si ésta es de radio unitario. Dicho centro se hace coincidir con el Aim Point.

parámetros de salida:

cosalfa, cosbeta, cosganma *coordenadas x, y, z de los vértices de los polígonos proyectados en la esfera (nº de helióstatos x nº de vértices)*

Cuando la esfera es de radio unidad estas coordenadas coinciden con los cosenos directores de las direcciones de proyección, de ahí que se hayan denominado así.

descripción:

? Las proyecciones de los vértices en la esfera se calculan como la intersección de la dirección de proyección, (que se fija como la que une los vértices girados de los espejos con el centro de la esfera (Aim Point)), y la superficie de la esfera dada por su radio unidad.

? Primero se obtiene la resta de las coordenadas del centro de la esfera y las de los vértices. Esta resta proporciona la dirección de proyección, a la que se le calcula el módulo y dividiendo por este último se consiguen *cosalfa, cosbeta, cosganma* que representan las coordenadas de los vértices en la superficie de la esfera de proyección.

Función planocm

Función llamada por **sombrasplano** (en caso de ser utilizada **bloqueosplano,** también la llamaría) y que calcula el punto de corte entre planos y rectas, en versión matricial.

```
function[Xcorte,Ycorte,Zcorte]=planocm(XRAY,YRAY,ZRAY,Cosray,XPLANE,YPLANE,
ZPLANE,Cosplano)
```

argumentos:

XRAY,YRAY,ZRAY	*matrices con las coordenadas x, y, z de los puntos que definen las rectas.*
Cosray	*cosenos directores de las rectas de las que se calcula la intersección con el plano (1 x 3)*
XPLANE,YPLANE,ZPLANE	*coordenadas del punto que define el plano del que se calcula la intersección*
Cosplano	*cosenos directores de la recta que define dicho plano (1x3)*

XRAY,YRAY,ZRAY son las mismas matrices definidas por XN, YN, ZN, calculadas en **vertices** y usadas en el cálculo de sombras y bloqueos. *Cosray* proporciona la dirección de las rectas que se interseccionan, que en el caso particular de cálculo de sombras son paralelas a la dirección del sol.

XPLANE,YPLANE,ZPLANE son calculadas en **sombrasplano**. Cosplano es el coseno director de la normal que define al plano.

parámetros de salida :

Xcorte, Ycorte, Zcorte	*coordenadas x, y, z de las intersecciones de las rectas con el plano*

Xcorte, Ycorte, Zcorte representarán las coordenadas de los todos los puntos de intersección que se calculen, expresadas en el mismo sistema de referencia que las rectas y el plano.

descripción:

? En esta función se realiza el cálculo de intersecciones de rectas, definidas por un punto y una dirección, con planos definidos de igual manera.

Es un cálculo general que sirve para cualquier posición relativa entre plano y recta; también es válido para cualquier nº de rectas de las que se quiera calcular la intersección; de hecho la principal ventaja del cálculo matricial es para un número elevado de intersecciones.

? En el uso que hace de **planocm, sombrastotal** se particularizan estas rectas y el plano:

? Las rectas son rayos paralelos a la luz del sol que incida en cada momento del día y que pasan por cada uno de los 4 vértices de los espejos; así habrá tantas rectas como *nº de helióstatos por 4*

? El plano está situado a una distancia suficientemente alejada y perpendicular a los rayos del sol

Las *(nº de helióstatos x 4)* intersecciones se calculan de forma matricial guardándose en Xcorte, Ycorte, Zcorte.

Función xcorteycorte

Función llamada por **sombrasplano** (en caso de ser utilizada, **bloqueosplano** también la llamaría); calcula los valores de x, y en el plano de proyección que corresponda en cada caso.

```
function[XCORTE,YCORTE]=xcorteycorte(XC,YC,ZC,XPLANE,YPLANE,ZPLANE,
Cosplano)
```

argumentos:

XC, YC, ZC	*coordenadas x, y, z de las intersecciones de las rectas con el plano de proyección (nº de helióstatos x nº de vértices)*
XPLANE, YPLANE, ZPLANE	*coordenadas del punto que define el plano en el que se calculan las intersecciones*
Cosplano	*cosenos directores de la recta que define dicho plano (1x3)*

XC, YC, ZC son idénticas a Xcorte, Ycorte, Zcorte calculados previamente por **planocm**. XPLANE, YPLANE, ZPLANE se toma como origen de coordenadas de un sistema que se dispone en el plano de proyección y al que se referirán todas las intersecciones.

parámetros de salida:

XCORTE, YCORTE	*coordenadas de los puntos de intersección rectas- plano, en el plano de proyección (nº de helióstatos x nº de vértices)*

descripción:

? Aunque **xcorteycorte** puede utilizarse de manera independiente para realizar un cambio de coordenadas, está asociada a **planocm** porque si se llaman consecutivamente ambas funciones, **xcorteycorte** utiliza como argumentos la salida de **planocm** y en cierta forma puede decirse que completa su tarea.

? **xcorteycorte,** a partir de los valores de corte con el plano de proyección expresados en el sistema de referencia 3D en que están referidos inicialmente el plano y las rectas, realiza un cambio de coordenadas a un nuevo sistema de referencia centrado en XPLANE, YPLANE, ZPLANE y contenido en el propio plano de proyección, es decir en un sistema plano.

La utilidad de expresar estas intersecciones en un mismo plano se debe a la necesidad de calcular las superposiciones que se produzcan entre los distintos polígonos proyectados, que representan los helióstatos. Para calcular estas intersecciones o solapamientos es necesario que los polígonos estén contenidos en el mismo plano.

Función solapevecinospm

Esta función es llamada por **sombrasplano** y por **bloqueosesfera**, aunque esta última lo hace indirectamente a través de **solapevecinosesf**. En caso de utilizarse, **bloqueosplano** también usaría **solapevecinospm.** Su misión es obtener solapamiento de polígonos.

```
[Areautil,parejasC,IintC] = solapevecinospm (X,Y,vecinos)
```

argumentos:

X, Y	*coordenadas de los vértices de los polígonos que solapan, en el plano de cálculo (nº de polígonos x nº de vértices)*
vecinos	*matriz que proporciona las parejas de vecinos; sus dimensiones dependen del nº de polígonos y del número de vecinos que se quiera calcular [(mxn)x2] 'm' es el número de polígonos y 'n' el de vecinos*

X, Y son calculadas por **xcorteycorte** y vecinos por **vecinos 3D**

parámetros de salida:

Areautil	*vector que indica el área del espejo "tras" el efecto coseno pero sin sombras (nº de helióstatos x 1)*
parejasC	*indica el orden de los polígonos que solapan, su tamaño es (nº de parejas que solapan x 2)*
IintC	*es el área de solapamientos en unidades X, Y su tamaño es el número de parejas*

Cuando *parejasC* e *IintC* son calculados cuando **solapevecinospm** es utilizada por **sombrasplano** o **bloqueosesfera** los polígonos representan los helióstatos y el área de solapamiento es el área de sombras o de bloqueos respectivamente.

descripción:

? Como se ha comentado antes, esta función calcula las superposiciones o solapamientos de polígonos regulares, dados por las coordenadas X, Y de sus vértices, siendo posible el algoritmo para cualquier número de vértices.

Otra particularidad es que esta función limita el cálculo de las superposiciones a los vecinos; los vecinos son los polígonos más cercanos a uno determinado del conjunto, el número de vecinos es fijado por el usuario. Si hacemos que el número de polígonos sea igual al de vecinos calcularíamos todas las posibles intersecciones.

De todas formas, no es probable cuando el grupo es numeroso, que los polígonos sean tan grandes como para que haya solapamientos de todos con todos, lo normal es que esto

ocurra entre un polígono dado y los que le rodean. Esta estrategia permite agilizar el tiempo de cálculo.

? En la aplicación de esta función al cálculo de la planta solar los solapamientos representarán las sombras y los bloqueos, según el caso.

Los polígonos representan los helióstatos, con lo que el número de vértices habitual será cuatro.

? El método de cálculo para los solapamientos entre dos polígonos cualesquiera es el siguiente:

Asignamos unos sentidos a los segmentos de ambos polígonos de forma que sigan el orden consecutivo según el cual vienen expresados en las matrices X, Y.

Para cada segmento del primer polígono, podemos calcular de una forma sencilla el solapamiento que se produce entre el área bajo el mismo y cada uno de los segmentos del segundo polígono, asignándole a cada una de estas áreas el signo positivo ó negativo según el sentido del primer segmento coincida ó no con el del segundo. Finalmente se suman todas estos solapamientos, y el valor absoluto de esta suma sería el solapamiento total de los dos polígonos escogidos. De esta forma hemos sustituido el cálculo complejo de una superposición por una serie de cálculos sencillos fáciles de computar:

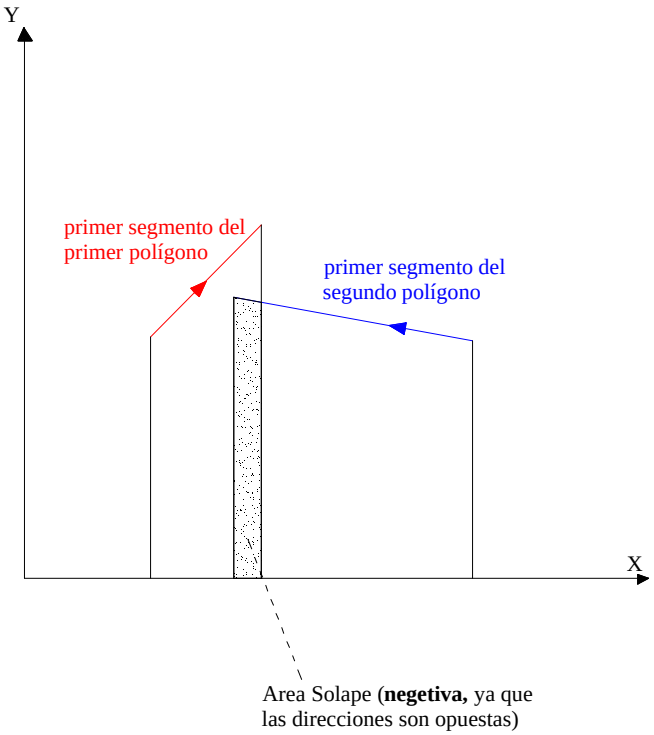


Figura 3.16: Se puede apreciar en esta figura el área de solape de dos de los segmentos de los polígonos. Se determinarían de manera similar estas áreas para todas las combinaciones posibles de segmentos de los polígonos.

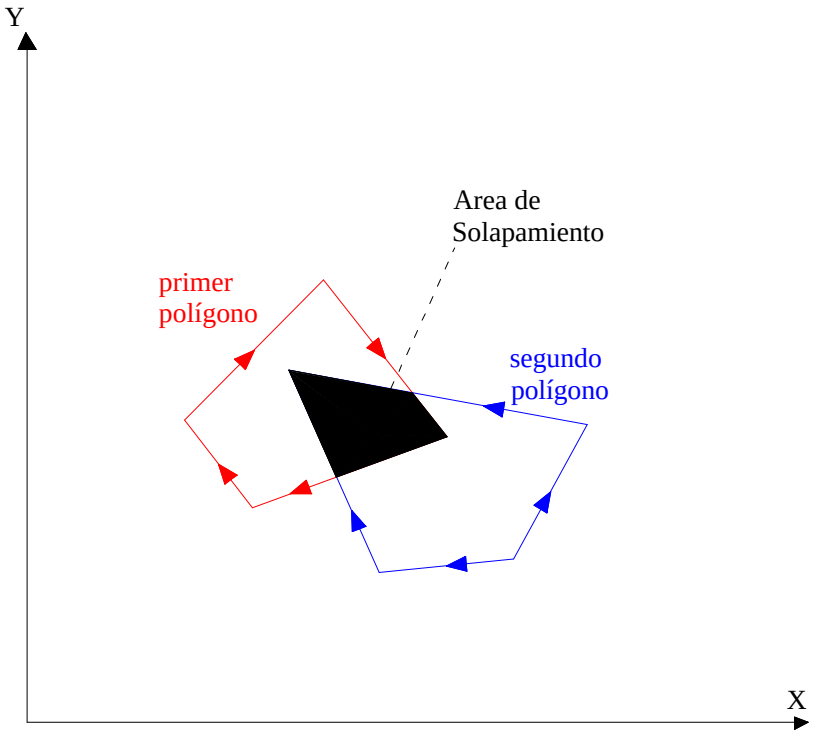


Figura 3.17: Aparece en negro la intersección de los dos polígonos. Es el resultado final del procedimiento.

? Finalmente expresamos los solapamientos en términos porcentuales respecto al área total del polígono que se encuentre bloqueado ó en sombra por su pareja.

Función solapevecinosesf

Función llamada por **bloqueosesfera** para calcular solapamientos de polígonos regulares sobre una esfera.

```
function[AreaM,ParejasC,IintC]=solapevecinosesf(XNSPH,YNSPH,ZNSPH,
   parejavecinos)
```

argumentos:

XNSPH,YNSPH,ZNSPH	*coordenadas x, y, z del centro de la esfera de proyección*
parejavecinos	*matriz* que *proporciona las parejas de vecinos;* *sus dimensiones dependen del nº de polígonos y del número de vecinos que se quiera calcular [(mxn)x2]' m' es el número de polígonos y 'n' el de vecinos*

XNSPH,YNSPH,ZNSPH definen la esfera de proyección, si ésta es de radio unitario. Dicho centro se hace coincidir con el Aim Point.

parejavecinos es calculada por **vecinos3D**

parámetros de salida:

AreaM	*vector que indica el área del polígono en la superficie de la esfera de proyección (nº de helióstatos x 1)*
parejasC	*indica el orden de los polígonos que solapan, su tamaño es (nº de parejas que solapan x 2)*
IintC	*es el área de solapamientos en unidades X, Y su tamaño es el número de parejas*

Cuando *parejasC* e *IintC* son utilizados por **bloqueosesfera** los polígonos representan los helióstatos y el área de solapamiento es el área de bloqueos

AreaM es calculada por la propia función **solapevecinosesf**

descripción:

? En esta función se realiza la proyección de polígonos regulares dados por sus vértices en la superficie de una esfera.

Para ello, los cosenos directores que definen la esfera se transforman mediante la función **azimelev** en coordenadas azimut, elevacion mediante las cuales se posicionará cualquier punto en la superficie de dicha esfera.

Después se llama a **solapevecinospm** que calculará los solapamientos como si la esfera fuese un plano.

? En el contexto del programa los polígonos son los helióstatos cuyos vértices se proyectan según la dirección que une el centro del espejo con el aim poin. La proyección se realiza en la esfera de radio unidad, pero centrada en el aim point

Sobre la superficie de dicha esfera es donde se calculan las superposiciones o solapamientos de los helióstatos proyectados, que representarán el bloqueo que sufre cada espejo.

Función rangoangular

Función llamada por **malla 3D**. Calcula el rango máximo de ángulos de mallado para cada valor del radio

```
function[thetaMax,thetaMin]=rangoangular(Pendientes,OrdenadasOrigen,
maxX,maxY,minX,minY,R,XV,YV)
```

argumentos:

Pendientes	*valor de las pendientes de los segmentos que forman el perímetro de la parcela o límite del terreno adquirido*
Ordenadasorigen	*valor de las ordenadas en el origen de cada una de los segmentos que delimitan la parcela adquirida*
maxX, maxY, minX, minY	*mayor y menor valor respectivamente de las coordenadas X e Y de los vértices de la parcela*
XV, YV	*vértices de la parcela en el plano inclinado*
R	*radio para el que se está calculando el intervalo angular*

Todos estos argumentos son calculados en la función **malla 3D** y son parámetros geométricos necesarios para delimitar la zona a mallar, es decir, así se marcan los límites para el cálculo de los espaciados radiales y azimutales.

parámetros de salida:

thetaMax, thetaMin	*valores máximo y mínimo que limitan el arco de circunferencia en el que se disponen los helióstatos para cada radio dado.*

Estos valores límite para la coordenada azimutal son los que delimitan el mallado, cuando el arco de circunferencia que representa cada radio dado corte a los segmentos límites más alejados de la parcela; a partir de estos valores, ya no se ponen helióstatos. En caso de parcelas con una disposición complicada será preciso eliminar algunos helióstatos sobrantes, lo que se realizará en la función **malla 3D.**

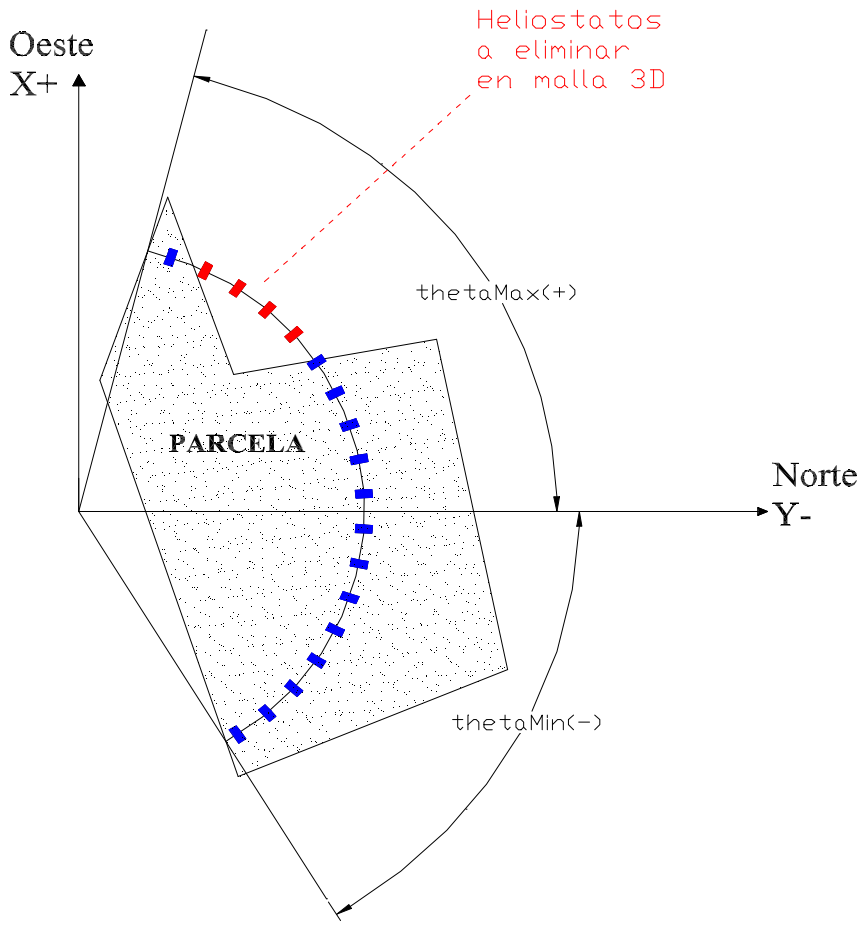


Figura 3.18: La función rangoangular delimita el arco de circunferencia en el que se dispondrán los helióstatos. Se calculan los valores de thetamax y thetamin que se aprecian en la figura y también como se eliminarían helióstatos intermedios.

descripción:

? Lo primero que hace esta función es llamar a **intercirseg** para calcular todas las posibles intersecciones entre los segmentos que forman la parcela, con circunferencias centradas en la torre, y de radio el calculado por **malla 3D**. Esta función proporciona los puntos de corte referidos a las coordenadas cartesianas usadas para situar el campo.

? Para el procedimiento de mallado es necesario cambiar estas coordenadas cartesianas a polares, para lo cual se usa la función de Matlab **cart2pol**. Además dado que los ejes cartesianos usados son tales que el campo está situado según la Y-(en dirección norte), y se quiere que theta 0 coincida con la Y- para que el proceso de mallado en la función **malla 3D** sea de más fácil tratamiento, entonces habrá que modificar la salida de **car2pol** para tener la referencia adecuada.

? Una vez hecho esto, se obtienen los valores de theta máximos y mínimos de los cortes.

? También se contemplan distintas casuísticas, que se puedan presentar por distintas posiciones particulares entre los segmentos y las circunferencias, debido a la forma de la parcela.

Función intercirseg

Función llamada por la función **rangoangular** para calcular intersecciones de segmentos con circunferencias.

```
function[corteX,corteY]=interCirSeg(Pendientes,OrdenadasOrigen,maxX,
maxY,minX,minY,R)
```

argumentos:

Pendientes	*valor de las pendientes de los segmentos de los que se va a calcular la intersección*
Ordenadasorigen	*valor de las ordenadas en el origen de cada una de los segmentos anteriores*
maxX, maxY, minX, minY	*mayor y menor valor respectivamente de las coordenadas X e Y de los extremos de los segmentos de los que se calcula la intersección*
R	*radio de la circunferencia de la que se está calculando la intersección con los segmentos*

Estos argumentos son exactamente los mismos que los de **rangoangular,** ya que esta función se los pasa directamente cuando llama a **intercirseg**. Los segmentos y circunferencias genéricas con las que trabaja **intercirseg** toman aquí el mismo significado que se le dio en **rangoangular**

parámetros de salida:

corteX, corteY	*obtenemos el/los valor/es de corte entre rectas y circunferencias dado/s por sus coordenadas X, Y*

Estos valores de corte, en el uso que se hace de **intercirseg**, son los que proporcionarán las coordenadas X, Y que proporcionarán, posteriormente, el rango de ángulos para limitar el mallado.

descripción:

? Esta función realiza las intersecciones de segmentos, contenidos en rectas que vienen dadas a su vez por su pendiente y su ordenada en el origen, con circunferencias de radio dado.

El procedimiento se divide en dos bloques: rectas verticales y rectas no verticales. Esto se debe a la especial complicación de las primeras al tener pendiente infinita, lo que originaría problemas si se siguiera la misma metodología que para las no verticales.

? Señalar que las rectas (que son infinitas) pueden cortar a una circunferencia en dos en uno o en ningún punto, y que además hay que evaluar, en caso de que haya corte, si este pertenece al segmento o no.

Función ordena

Función llamada por **malla 3D** para ordenar el campo de helióstatos calculado por el método de los no bloqueos

```
function[CAMPO,POSICIONX12]=ordena(X, Y, Z)
```

argumentos:

X, Y ,Z	*matrices (m x n) con las coordenadas de los pedestales del campo de helióstatos*

Coordenadas calculadas por **malla 3D,** pasadas por parámetro, en las que puede haber algún helióstato nulo. *m* es el número de anillos y *n* es el número máximo de helióstatos por anillo

parámetros de salida:

CAMPO	*matriz de tamaño (N x 3) con las coordenadas de los pedestales*
POSICIONX12	*matriz de dimensión (N x 2) en la que se asigna una posición a cada helióstato*

En *CAMPO* se han unido las tres matrices de las coordenadas *X ,Y ,Z* en una sola, con el fin de obtener mayor facilidad para operar con ella. N es por tanto el número total de helióstatos.

POSICIONX12 es una matriz en la que se guarda en que anillo está el helióstato y que posición ocupa dentro de este.

descripción:

? La función **malla 3D** genera las coordenadas x, y, z de los centros de los helióstatos y las almacena en tres matrices, una para cada clase de coordenada. Cada una de estas matrices tienen tantas filas como número de anillos en el campo y tantas columnas como el número máximo de helióstatos en cada anillo; esto hace que haya posiciones en la matriz que no se correspondan con helióstatos reales, por ejemplo, si en el anillo de máximo número de espejos hubiese 15 y en otro 10, las matrices *X, Y, Z* tendrían 15 columnas y por tanto en la fila de 10 habría cinco posiciones que no se corresponden con helióstatos, a las que se asigna el valor cero.

? Las coordenadas del campo colector son usadas numerosas veces en muchas funciones del programa, por ello hay que guardarlas de una forma más eficiente. **ordena** toma las tres matrices y las convierte en una sola, *CAMPO* que está ordenada de forma que los algoritmos que la usan funcionan más rápido a como lo harían con las otras tres. Adicionalmente se han eliminado las posiciones no válidas de dichas matrices iniciales, en las cuales aparecían valores nulos inservibles, que de otra forma arrastraríamos inútilmente a otras operaciones.

? Por último, comentar que la matriz POSICIONX12, ayuda a identificar los helióstatos porque en CAMPO no aparecen ordenados por filas y posición en la fila, sino en el orden en que se han ido poniendo.

Función ispar

Función llamada por **malla 3D** que permite saber si el número introducido como argumento es par o impar.

```
function[espar]=ispar(integer)
```

argumentos:

integer — *número entero cualquiera*

El número entero será el número de helióstatos en un anillo o el número de anillos en un grupo.

parámetros de salida:

espar — *variable que toma valor 0 o 1 en función de que el número sea par o impar*

Si *espar* toma el valor 1, el entero introducido es par y si toma el valor 0 será impar

descripción:

? Esta función ha sido necesaria para realizar algunas de las condiciones del procedimiento de mallado para las que es necesario distinguir, si el numero del anillo de un grupo es par ó impar puesto que no se realizan las mismas operaciones con uno ú otro caso. Esta diferencia se basa en que en el primer anillo de cada grupo y los posteriores impares hay un helióstato situado en el eje Y-, y debido a que el mallado se realiza barriendo primero los valores de **theta** positivos, y luego los negativos, a partir de dicho eje, hay un helióstato en él que es necesario asignar, arbitrariamente, a una de las dos ramas. En los anillos pares del grupo no se coloca ningún helióstato en el eje Y-, por lo que no será necesario realizar esta asignación.

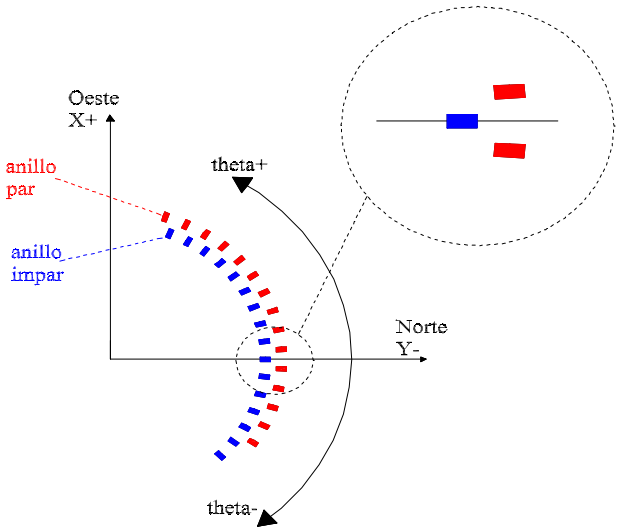


Figura 3.19: Situación de anillo par (en rojo) e impar (en azul).

Función costetorre

Función llamada en **optimiza 3D** para calcular el coste de la torre

```
function [TORRECOST]=costetorre(THT)
```

argumentos:

THT	*altura de la torre*	*(metros)*

THT con la que se modela el coste de la torre es la que haya proporcionado el proceso de optimización, es decir la altura elegida para el campo óptimo.

parámetros de salida:

TORRECOST	*coste de la torre*	*(€)*

descripción:

? Se calcula el coste de la torre en función de su altura *THT*. Se observa que lógicamente el coste aumenta en función de la altura. Este aumento es prácticamente cuadrático.

$$\cos teEUR = 1445783.133 \times \left(THT / 150 \right)^{1.5}$$

Función costeterreno

Función llamada por **optimiza 3D** para calcular el coste del terreno

```
function[TERRENOCOST]=costeterrenoMio(AreaEnSuelo,PENDIENTETERRENO)
```

argumentos:

AreaEnSuelo	*área de la parcela en el terreno inclinado, proyectada sobre el plano horizontal (m²)*
PENDIENTETERRENO	*ángulo de inclinación del terreno respecto a la horizontal (grados)*

parámetros de salida:

TERRENOCOST	*coste del terreno (€)*

descripción:

? EL coste del terreno dependerá no solo del área del mismo, sino también de otros factores como pueden ser las distintas operaciones (movimientos tierras,…) necesarias para poder finalmente tener una superficie en forma de pendiente plana ó de cono. En nuestro programa se contempla esta posibilidad mediante la función *f(PENDIENTETERRENO)*, sin embargo no se han tenido en cuenta dichas circunstancias al hacer *f=1,* y solo relacionamos el coste con el área de la parcela proyectada sobre el plano horizontal:

$$TERRENOCOSTE = f(PENDIENTETERRENO) \times AreaEnSuelo \times 5 euros / m^2$$

donde $f(PENDIENTETERRENO) = 1$ en este caso.

Función costereceptor

Función llamada por **optimiza3D** para calcular el coste del receptor

```
function [RECCOST]=costerec(Areareceptor)
```

argumentos:

Areareceptor *área del receptor* *(m^2)*

El área del receptor se ha modelado como la de una circunferencia, a partir de su radio. Se trata pues, de un receptor externo, plano. Éste área se refiere al área de los paneles de absorción.

parámetros de salida:

RECCOST *coste del receptor* *(€)*

descripción:

? Se calcula el coste del receptor en función de su área con la siguiente fórmula:

$$RECCOST = 542168.67 \times \left(Areareceptor / 92.85 \right)^{0.8}$$

donde 542.168.67 €

El coste del receptor será constante en el proceso de optimización al modelarse como función del área, ya que ésta no varía en este proceso

Función pintasolucion

Función llamada por **optimiza 3D** para obtener gráficamente los resultados y obtener el fichero de datos de salida resultante del proceso de optimización

```
function
pintasolucion3D(SOLUCION,SOLUCIONDESIGN,ALTURAH,ANCHURAH,angulonuevo,CRITERIO
,XV,YV,cosinc,PEDESTH)
```

argumentos:

SOLUCION	*estructura con los datos medios anuales en los campos calculados*
SOLUCIONDESIGN	*estructura con los datos del punto de diseño en los campos calculados*
ANCHURAH, ALTURAH	*dimensiones de los helióstatos y altura del*
PEDESTH	*pedestal*
CRITERIO	*elección del tipo de optimización*
XV,YV	*vértices de la parcela*
angulonuevo	*inclinación del terreno respecto a la horizontal*
cosinc	*cosenos directores de los rayos incidentes*

SOLUCION, SOLUCIONDESIGN han sido calculados en **optimiza3D,** y *cosinc* por **cosdirinc.** Los demás argumentos son datos fijados por el usuario y pasados por parámetros.

parámetros de salida:

Esta función no devuelve ningún dato como los de las demás funciones, pero proporciona ficheros y figuras con los datos de salida del programa que se han considerado más relevantes.

descripción

? No explicaremos en detalle la secuencia que sigue esta función ya que básicamente utiliza otras funciones existentes de MATLAB como son:

→ Para la representación de figuras: *figure, fill3, colormap, colorbar*

→ Para la elaboración de ficheros de datos: *fprintf, fopen*

? En primer lugar la función devuelve una serie de figuras (*ver Anexo II*) consistentes en los helióstatos del campo óptimo calculado, rellenos según una escala de colores que se corresponderá con la magnitud del dato representado en cada una de las figuras. En concreto habrá figuras para los datos siguientes:

Potencia media anual en kW
Pérdidas medias anuales por efecto coseno en kW
Pérdidas medias anuales por sombras del terreno en kW
Pérdidas medias anuales por sombras en kW
Pérdidas medias anuales por bloqueos en kW
Pérdidas medias anuales por atenuación atmosférica en kW
Pérdidas medias anuales por spillage en kW

En todas ellas se dibuja cada helióstato con una orientación estándar elegida, ya que al representar los datos de medias anuales no tiene sentido la orientación del helióstato para una dirección incidente del sol concreta.

Una última figura expone la Potencia de cada helióstato en el punto de diseño en kW, y ahora al ser la dirección incidente del sol una determinada, cada helióstato se ha dibujado con su orientación real, además de incluir para cada uno de ellos el rayo incidente, el reflejado y la bisectriz de éstos (que es perpendicular al plano del helióstato). También se dibujan los pedestales y la torre del receptor, todo ello para que la figura dé una idea real de la planta en su conjunto.

? En segundo lugar la función elabora dos ficheros de datos (*ver Anexo III*), uno para las medias anuales y otro para el punto de diseño, en los cuales se representa para cada helióstato del campo óptimo:

Potencia en kW
Pérdidas por efecto coseno en kW
Pérdidas por sombras del terreno en kW
Pérdidas por sombras en kW
Pérdidas por bloqueos en kW
Pérdidas por atenuación atmosférica en kW
Pérdidas por spillage en kW